

КОМПЬЮТЕРЫ

ПРАВОВОЕ РУКОВОДСТВО

В ТРЕХ ТОМАХ

3

Под редакцией Г. Хелмса

**Перевод с английского
под редакцией
канд. техн. наук В. Г. Потемкина**

Scan Pirat

**МОСКВА «МИР»
1986**

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА	5
Глава 20. МАШИННАЯ ГРАФИКА <i>У. Ньюмен. Р. Спрулл</i>	7
20.1. Введение	7
20.2. История развития машинной графики	11
20.3. Как работают интерактивные графические дисплеи	12
20.4. Некоторые общие вопросы	14
20.5. Новые типы дисплеев	17
20.6. Универсальное программное обеспечение машинной графики	18
20.7. Организация взаимодействия с пользователем	19
20.8. Построение изображений твердых тел	20
20.9. Техника растровых изображений	22
20.10. Система координат	23
20.11. Метод приращений	24
20.12. Алгоритмы вычерчивания отрезков	25
20.13. Генераторы окружностей	33
20.14. Векторные дисплеи	35
20.15. Дисплеи и дисплейные контроллеры	37
20.16. Дисплеи	37
20.17. Конструкция ЭЛТ	38
20.18. Графические устройства с запоминанием изображения	45
20.19. Дисплеи с запоминающей трубкой	51
20.20. Векторные дисплеи с регенерацией изображения	54
Глава 21. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И РОБОТОТЕХНИКА. <i>Г. Хелмс</i>	60
21.1. Введение	60
21.2. Компьютер и интеллект	60
21.3. Экспертные системы	61
21.4. Робототехника	63
21.5. Задачи в робототехнике	63
21.6. Роботы и персональные компьютеры	64
Глава 22. СИМВОЛЬНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА. <i>Л. Хонстейн</i>	66
22.1. Введение	66
22.2. Классификация печатающих устройств	67
22.3. Набор символов	69
22.4. Посимвольные ударные печатающие устройства с цельным очертанием символов	70

Оглавление

22.5.	Построчные ударные печатающие устройства с цельным очертанием символов	76
22.6.	Построчные безударные печатающие устройства с цель- ным очертанием символов	79
22.7.	Точечно-матричные печатающие устройства	82
22.8.	Посимвольные ударные точечно-матричные печатающие устройства	85
22.9.	Построчные ударные точечно-матричные печатающие устройства	89
22.10.	Безударные матричные печатающие устройства	90
22.11.	Подвод бумаги	96
Глава 23.	ГРАФОПОСТРОИТЕЛИ Л. Хознстейн	99
23.1.	Введение	99
23.2.	Методы формирования изображения	99
23.3.	Перемещение бумаг	103
23.4.	Конструкции графопостроителей	106
23.5.	Технические характеристики графопостроителей	107
23.6.	Применение графопостроителей	108
23.7.	Графические программы	109
Глава 24.	ОБЗОР МИКРОПРОЦЕССОРНОЙ ТЕХНОЛОГИИ. Д. Стоут	119
24.1.	Эволюция микропроцессоров	119
24.2.	Компоненты микропроцессора	122
24.3.	Программное обеспечение микропроцессоров и микроЭВМ	131
24.4.	Основы проектирования микропроцессорных систем	139
	Литература	142
Глава 25.	МИКРОЭВМ И ПРОГРАММИРОВАНИЕ. К. Виатровски, Ч. Хаус	143
25.1.	Введение	143
25.2.	Описание программ	147
25.3.	Типы данных	152
25.4.	Хранение данных	155
25.5.	Команды	156
25.6.	Организация микроЭВМ на базе микропроцессоров 8080/8085 фирмы Intel	157
25.7.	Язык ассемблера	162
25.8.	Команды пересылки данных	166
25.9.	Команды обработки булевых данных	174
25.10.	Команды циклического сдвига	177
25.11.	Команды перехода	181
25.12.	Пример логического устройства управления	188
25.13.	Другой подход к построению логического устройства управления (контроллера)	196
Глава 26.	ПОДПРОГРАММЫ, ПРЕРЫВАНИЯ И АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ К. Виатровски, Ч. Хаус	200
26.1.	Подпрограммы	200
26.2.	Прерывания	206
26.3.	Дополнительные псевдокоманды	213
26.4.	Пример реализации контроллера, действующего в полу- автоматическом режиме	215
26.5.	Арифметические операции	217

26.6.	Дополнительные команды пересылки данных	232
26.7.	Программируемые арифметические операции	234
Глава 27.	ПОНЯТИЕ ИНТЕРФЕЙСА. Д. Гивоне, Р. Россер	248
27.1.	Порты ввода-вывода	248
27.2.	Координация взаимодействия с внешними устройствами	253
27.3.	Прерывания программы	260
27.4.	Интерфейс с главной памятью	272
27.5.	Прямой доступ к памяти	284
27.6.	Дополнительные сведения о шинах	289
27.7.	Цифро-аналоговые преобразователи	292
27.8.	Последовательный ввод-вывод	308
27.9.	Секционированные микропроцессоры	316
27.10.	Синхронизация микропроцессоров	325
Глава 28.	ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ МИКРОКОМПЬЮТЕ- РОВ. С. Мясковски	328
28.1.	Введение	328
28.2.	Компоненты операционных систем	328
28.3.	Особенности операционных систем для микрокомпьютеров	329
28.4.	Микрокомпьютеры против универсальных ЭВМ	330
28.5.	Проблемы совместимости и мобильности операционных систем	331
28.6.	Операционная система CP/M	332
28.7.	Усовершенствованные операционные системы для 8-разрядных компьютеров	337
28.8.	Операционные системы для 16-разрядных микропроцессоров	339
28.9.	Заклучение	341
Глава 29.	ВЫВОД ЗВУКОВОЙ ИНФОРМАЦИИ: ВОСПРОИЗВЕДЕНИЕ МУЗЫКИ И РЕЧИ. Л. Хознстейн	342
29.1.	Введение	342
29.2.	Звуковые ответчики	342
29.3.	Синтезаторы музыки и шумовых эффектов	344
29.4.	Синтезаторы речи	345
Глава 30.	СИСТЕМА РАСПОЗНАВАНИЯ РЕЧИ. Ф. Коперда	358
30.1.	Введение	358
30.2.	Цели проекта	359
30.3.	Общая характеристика системы	360
30.4.	Первичная обработка речевых сигналов	362
30.5.	Сжатие информации	365
30.6.	Определение границ слов	368
30.7.	Объяснение некоторых лингвистических терминов	372
30.8.	Выделение звуковых образов	374
30.9.	Параметры образов и их анализ	377
30.10.	Методика идентификации слов	387
30.11.	Аппаратная реализация	388
30.12.	Текущее состояние проекта	390
	Литература	391
	Список терминов	392
	Предметный указатель	400

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

Последний том справочного руководства по компьютерам включает главы, освещающие прикладные проблемы использования средств вычислительной техники. Дается обзор и описываются принципы работы различных типов графических дисплеев, печатающих устройств и графопостроителей, рассматриваются области наиболее эффективного применения этих устройств.

В главах, посвященных микрокомпьютерам, отмечены основные этапы развития микропроцессорной техники, проанализированы принципы функционирования отдельных устройств микроЭВМ, рассмотрены наиболее важные элементы их программного обеспечения. Значительное внимание уделено вопросам программирования. Очень удачно с методической точки зрения построена глава, описывающая полную систему команд микропроцессора Intel 8085. Знакомства с материалом этой главы вполне достаточно для приобретения навыков программирования на микроЭВМ. Примеры, иллюстрирующие различные приемы программирования, связаны с решением реальных задач. Изложение вопросов построения микропроцессорных систем завершается обзором операционных систем для микроЭВМ. Этот материал изложен кратко и носит сугубо справочный характер.

Представляют безусловный интерес рассмотренные в томе проблемы искусственного интеллекта, робототехники, воспроизведения с помощью ЭВМ музыки и речи, распознавания речевых сигналов. Приведены примеры конкретных систем, перечислены основные идеи, применяемые для их создания. Хотя в материале по распознаванию речи используются примеры из английской фонетики, он носит методический характер и не зависит от конкретного языка.

В целом том охватывает широкий спектр вопросов, связанных с использованием вычислительных средств в различных технических областях. Это, наряду с фактическими справочными данными по характеристикам компьютеров и их

программному обеспечению, раскрывает области применения вычислительной техники и дает перспективу ее развития.

Книга может быть рекомендована инженерам, аспирантам и студентам как справочное руководство.

Перевод тома выполнили: канд. техн. наук А. В. Шалашов (гл. 20, 29, 30), А. В. Чукашев (гл. 21, 28), В. В. Пржиялковский (гл. 22, 23), канд. техн. наук М. В. Сергиевский (гл. 24, 25), канд. техн. наук И. П. Пчелинцев (гл. 26), канд. физ.-мат. наук В. С. Штаркман и Т. А. Шаргина (гл. 27).

В. Г. Потемкин

20

МАШИННАЯ ГРАФИКА¹⁾

У. Ньюмен, Р. Спрулл

20.1. ВВЕДЕНИЕ

В настоящее время широкую популярность приобрели телевизионные игры — сравнительно недавнее изобретение для домашнего досуга. Принцип действия одной из таких игр, напоминающей настольный теннис, продемонстрирован на рис. 20.1. В нее играют двое, пользуясь парой рычажков-регуляторов, подключенных к обычному бытовому телевизору. После включения игры по экрану взад-вперед начинает бегать маленькое светящееся пятно, имитирующее мяч. Каждый из игроков с помощью своего рычажка старается так переместить «ракетку», чтобы мяч, отразившись от нее, перелетел на противоположную сторону «площадки». Если игрок пропускает мяч, его партнеру дается одно очко. Выигрывает тот, кто первым наберет 15 очков.

Телевизионные игры стали первым примером бытового применения **машинной графики**, т. е. методов создания и преобразования изображений с помощью ЭВМ. Такие изображения можно воспроизводить на бумаге или пленке, используя **графопостроители**, управляемые ЭВМ. Машинная графика нашла применение и как средство художественного оформления, в частности для синтеза заставок, появляющихся перед началом телевизионных передач (рис. 20.2). Все изображения подобного типа получают с помощью **пассивной**, или **неинтерактивной** машинной графики, которая не предоставляет зрителю возможности вмешиваться в действия ЭВМ. Чтобы зритель мог манипулировать изображением, ему необходимо иметь какое-то устройство ввода информации, например в описанной выше игре в настольный теннис им служит рычажок-регулятор, который вырабатывает управляющие сигналы, передаваемые в ЭВМ. С введением таких устройств машинная графика становится **интерактивной**.

¹⁾ Adapted from Principles of Interactive Computer Graphics, second edition, by William M. Newman and Robert F. Sproull. Copyright © 1979. Used by permission of McGraw-Hill, Inc. All rights reserved.

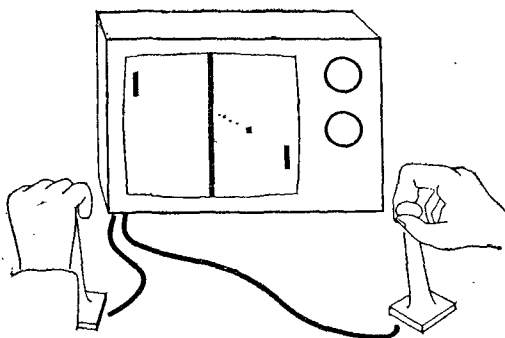


Рис. 20.1. Машинная графика на дому: телевизионная игра в настольный теннис.

Интерактивная машинная графика предусматривает двусторонний обмен информацией между ЭВМ и пользователем. После того как получен сигнал от входного устройства, ЭВМ соответствующим образом преобразует генерируемое изображение. Пользователь же воспринимает это просто как мгновенное изменение наблюдаемой им картинке в ответ на поданный сигнал. Он может ввести целую последовательность команд, каждая из которых вызывает определенную реакцию ЭВМ, которая соответствующим образом трансформирует изображение. Действуя таким способом, пользователь ведет с ЭВМ своего рода разговор, или диалог.



Рис. 20.2. Телевизионная заставка, синтезированная ЭВМ (с разрешения фирмы Information International, Inc.).

Среди множества применений интерактивной машинной графики есть и такие, которые имеют отношение к обеспечению безопасности нашей жизни. В частности, графические системы используются при подготовке пилотов гражданских авиалиний. Значительную часть тренировочного времени пилоты проводят не в воздухе, а на земле за органами управления **авиатренажера**. Такой тренажер представляет собой выполненный в натуральную величину макет кабины летчиков со всеми штатными приборами и устройствами. Кроме того, в кабину встроены экраны дисплеев, на которых воспроизводятся генерируемые ЭВМ изображения местности, наблюдаемой пилотом в процессе взлета или посадки самолета. Когда тренирующийся летчик начинает манипулировать органами управления, эти изображения меняются, создавая полную иллюзию движения самолета. Применение авиатренажеров при подготовке пилотов имеет целый ряд преимуществ по сравнению с проведением реальных полетов. Благодаря им обеспечивается экономия топлива, устраняется риск катастрофы, создается возможность ознакомления летчиков с особенностями многих местных и зарубежных аэропортов.

Еще более важную роль интерактивная машинная графика играет в электронной промышленности. Количество компонентов в стандартной интегральной электронной схеме типа тех, что используются в ЭВМ, столь велико, что инженеру требуется несколько недель, чтобы вычертить ее вручную. Не меньше времени уходит и на перечерчивание, если в схему вносятся серьезные изменения. За счет внедрения интерактивных графических систем, подобных изображенной на рис. 20.3, затраты времени на получение чертежей сокращаются во много раз. С помощью ЭВМ инженер может, кроме того, осуществлять проверку спроектированной схемы и вносить необходимые коррективы, причем эти операции занимают считанные минуты. Следует отметить, что постоянная тенденция к снижению стоимости электронной аппаратуры в существенной мере объясня-

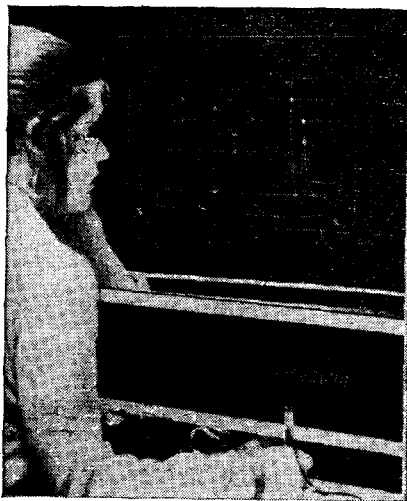


Рис. 20.3. Проектирование электронной схемы на установке, включающей планшет и графический дисплей (с разрешения фирмы Calma, филиала фирмы General Electric).



Рис. 20.4. Примеры использования интерактивной графики для вывода планов и схем в архитектуре и градостроении.

ется именно прогрессом средств проектирования интегральных схем.

Было приведено лишь два примера промышленного использования интерактивной машинной графики, позволяющей решать технические задачи, реализация которых другими способами потребовала бы чрезмерно больших расходов. Однако имеется и множество других проблем, решаемых с помощью интерактивной графики либо более эффективно, либо с меньшими затратами. Например, архитектор, работая с интерактивным графическим терминалом, получает возможность сравнивать различные варианты проекта; при этом он может проанализировать множество альтернативных решений, что без ЭВМ ему сделать было бы не под силу. Ученый, занимающийся молекулярной биологией, может выводить на экран изображения молекул и исследовать их внутреннюю структуру. Строители городов и транспортных магистралей могут пользоваться ЭВМ для построения карт и планов, снабженных дополнительной информацией, необходимой при проведении планировочных работ. Некоторые из рассмотренных приложений интерактивной графики показаны на рис. 20.4.

Основной выигрыш от применения интерактивной машинной графики заключается в удобстве и повышенной скорости восприятия пользователем ЭВМ той информации, которая выводится для него на экран дисплея. Например, инженер, проектирующий интегральную схему, может заметить на экране такие ее особенности, которые он почти наверное пропустил бы, просматривая обычную распечатку с цифровыми данными. Проектировщик, имея возможность взаимодействовать с ЭВМ, может быстро внести требуемые изменения и вновь вывести на экран скорректированный чертеж схемы. Итак, мы можем

сделать вывод, что интерактивная графика позволяет расширить **полосу пропускания** информационного канала, через который осуществляется двусторонняя связь между пользователем и ЭВМ.

20.2. ИСТОРИЯ РАЗВИТИЯ МАШИННОЙ ГРАФИКИ

Для того чтобы все это стало возможным, потребовались годы интенсивных исследований и конструкторских работ. Впервые простейшие изображения на дисплее, соединенном с ЭВМ, удалось получить в 1950 г. в Массачусетском технологическом институте (МТИ) на машине Whirlwind I. В дисплее была применена **электронно-лучевая трубка (ЭЛТ)**, подобная устанавливаемым в бытовых телевизорах. Несколькими годами раньше покойный Ф. Уильямс на базе ЭЛТ разработал устройство для хранения информации; впоследствии трубки такого типа вновь получили распространение, но уже в качестве запоминающих ЭЛТ, которыми оснащаются многие модели интерактивных графических терминалов, отличающиеся невысокой стоимостью.

На протяжении 50-х годов интерактивная машинная графика развивалась относительно медленно, поскольку ЭВМ того периода не были приспособлены к работе в диалоговом режиме. Они представляли собой просто быстродействующие арифмометры, использовавшиеся физиками и разработчиками ракетной техники для проведения трудоемких вычислений. Лишь к концу десятилетия с появлением ЭВМ типа TX-0 и TX-2, созданных в МТИ, проблемы реализации диалога существенно упростились, и это обстоятельство способствовало быстрому росту интереса к машинной графике.

Событием, которое сразу выдвинуло интерактивную машинную графику в качестве новой важной сферы науки и техники, стала публикация в 1962 г. блестящей диссертационной работы Сазерленда, удостоенного степени доктора философии в МТИ. Эта диссертация, озаглавленная «Скетчпэд: человеко-машинная графическая система связи», доказала многим читателям, что интерактивная машинная графика — это перспективная, имеющая большое практическое значение и, что немало важно, весьма интересная область исследований. Уже к середине 60-годов работы по созданию больших машинных графических систем быстрыми темпами велись в МТИ, в фирмах «Дженерал моторз», «Белл телефоун лэбораториз», «Локхид эйркрафт» и других. Так начался золотой век машинной графики.

Если в 60-х годах машинная графика в основном служила предметом теоретических исследований, то 70-е годы стали десятилетием, в котором эти работы начали приносить первые

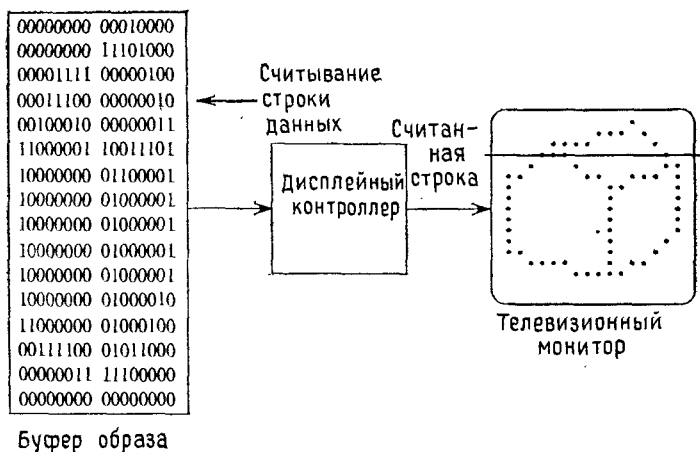


Рис. 20.5. Схема работы дисплея, оснащенного буфером образа.

практические результаты. В настоящее время интерактивные графические терминалы получили широкое распространение во многих странах мира. Они успешно применяются в системе образования, в том числе и в обычных средних школах. Простота освоения средств машинной графики пользователями, относящимися к самым разным возрастным группам, способствует ее проникновению в разнообразные прикладные сферы и, несомненно, обеспечивает дальнейший рост ее популярности.

20.3. КАК РАБОТАЮТ ИНТЕРАКТИВНЫЕ ГРАФИЧЕСКИЕ ДИСПЛЕИ

Устройство современного графического дисплея отличается чрезвычайной простотой. Он состоит из трех основных блоков: цифровой памяти, или **буфера образа**, для хранения закодированного изображения, представленного в форме матрицы значений интенсивностей свечения точек экрана; телевизионного монитора, который отличается от обычного телевизора лишь отсутствием цепей приема и настройки; наконец, блока сопряжения, именуемого **дисплейным контроллером**, который обрабатывает и передает в монитор содержимое буфера образа. Для того чтобы картина на экране дисплея воспринималась пользователем как установившееся, не мерцающее изображение, она должна постоянно возобновляться с частотой, не меньшей 30 раз в секунду.

В буфере образа изображение хранится в виде двумерного массива двоичных чисел; они кодируют простейшие, неделимые

элементы изображения, называемые **пикселями**. В простейшем случае, когда необходимо воспроизвести лишь черно-белый вариант изображения без градаций оттенков, черные пиксели могут быть представлены, например, единицами, а белые — нулями. На рис. 20.5 показано, как массив черных и белых пикселей, образующих квадрат 16×16 , можно записать набором из 32-х 8-разрядных двоичных чисел.

Дисплейный контроллер последовательно байт за байтом считывает данные из буфера образа и преобразует двоичные единицы и нули в соответствующие видеосигналы. Эти сигналы подаются далее на телевизионный монитор, в результате чего на его экране возникает изображение, состоящее из черных и белых точек. Пример подобного изображения представлен на рис. 20.6. Описанную процедуру дисплейный контроллер повторяет не менее 30 раз в секунду, благодаря чему пользователь не замечает пульсации изображения на экране телевизионного монитора.

Предположим, что нам понадобилось вывести на экран новое изображение. Все, что в этом случае нужно сделать, — это соответствующим образом изменить содержимое буфера образа, чтобы оно отображало требуемую совокупность пикселей. Таким способом можно, например, показать вращающееся колесо или же колесо, которое попеременно то растягивается, то сжимается.

Теперь нетрудно понять, как можно запрограммировать игру, имитирующую настольный теннис. Каждое из 16 возможных положений ракетки, принадлежащей правому игроку, кодируется определенным

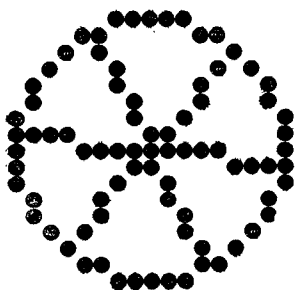


Рис. 20.6. Растровое изображение колеса.

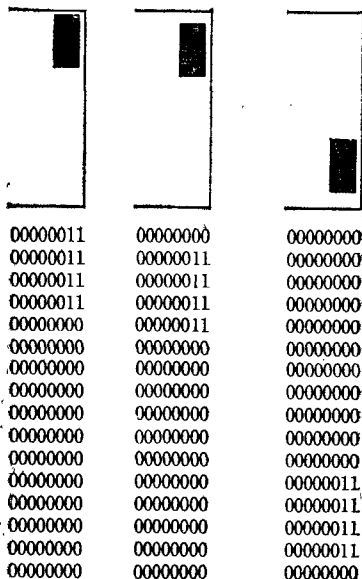


Рис. 20.7. Содержимое буфера образа для трех из 16 возможных положений ракетки.

набором двоичных цифр; некоторые из них приведены на рис. 20.7. В процессор поступает код, определяющий текущую позицию правого рычажка-регулятора; в ответ процессор формирует соответствующий массив чисел и засылает его в правую половину буфера образа, занимающую 16 байт. Аналогичные операции выполняются с левым рычажком и левой половиной буфера. Далее вычисляется положение «мяча» и в буфере образа соответствующий бит устанавливается в «1». Описанный процесс повторяется вновь и вновь; при этом дисплейный контроллер всякий раз передает содержимое буфера в телевизионный монитор, на экране которого воспроизводится динамично меняющаяся картина.

20.4. НЕКОТОРЫЕ ОБЩИЕ ВОПРОСЫ

Чтобы не создалось впечатления, что те, кто затратили годы труда и огромные средства на исследования по машинной графике, добились слишком скромных результатов, необходимо подчеркнуть, что реализация интерактивного графического режима требует решения множества проблем, не упомянутых при обсуждении предыдущего примера. Возможно, у некоторых читателей уже возникли вопросы по поводу приведенного примера. В этом разделе мы сформулируем некоторые из наиболее часто задаваемых относительно машинной графики вопросов и попытаемся дать на них ответ.

Как выводятся на экран прямые линии? Как вычерчиваются на дисплее кривые?

Две проблемы, возникающие при воспроизведении с помощью графического дисплея прямых и кривых линий, можно проиллюстрировать на примере колеса, изображенного на рис. 20.6. Во-первых, необходимо установить, какие пиксели будут черными и какие белыми; данный выбор отнюдь не всегда очевиден. Во-вторых, идущие с наклоном прямые и кривые линии на приведенной картинке далеки от гладкости и вместо этого воспроизводятся в виде режущих глаз «ступенек».

Для решения первой из указанных проблем используется процедура (или **алгоритм**), которая (ый) по уравнению прямой или кривой устанавливает те пиксели, которые должны быть черными. Большинство таких алгоритмов столь просты, что их без труда можно реализовать аппаратными средствами и тем самым резко повысить скорость генерации линий.

Значительно сложнее решается проблема, обусловленная **квантованием** и проявляющаяся в виде ступенчатых изображений. Чаще всего ее устраняют путем использования устройств

другого типа, именуемых **векторными дисплеями**, на которых изображения строятся не из отдельных точек, а из соединенных друг с другом отрезков прямых и кривых. Имея векторный дисплей, можно вычерчивать линии, которые невооруженному глазу будут казаться совершенно гладкими.

Вплоть до недавнего времени из всего множества созданных графических терминалов широко применялись только векторные дисплеи; внедрению дисплеев с буферами образов препятствовала высокая стоимость элементов цифровой памяти. И хотя теперь ситуация существенно изменилась, основные исследования по машинной графике ориентированы на использование именно векторных дисплеев; вопросы построения буферов образа и эффективной реализации их возможностей нуждаются в дальнейшем изучении.

Почему при воспроизведении изображений столь большое значение придается быстродействию ЭВМ?

И здесь вновь можно указать две причины. Во-первых, любой дисплей на базе ЭЛТ нуждается в регенерации, которая осуществляется путем периодической засылки в него закодированного изображения. Изображение поступает в дисплей поэлементно, либо точка за точкой, либо отрезок за отрезком (в случае векторного дисплея). Если все изображение целиком будет передаваться с частотой, меньшей 25 Гц, то на экране возникнет неприятное для глаз смотрящего мерцание. Чем больше времени занимает передача каждого элемента изображения, тем меньше может быть общее число передаваемых элементов и, следовательно, тем меньше объем выводимой информации. Дисплеи ранних разработок, подобные использованному Сазерлендом при создании системы **Sketchpad**, давали возможность выводить без мерцания всего несколько сотен точек; на экранах современных векторных дисплеев одновременно могут вычерчиваться многие тысячи отрезков, и тем не менее мерцание отсутствует.

Второй аспект проблемы быстродействия связан с продолжительностью **реакции** машинной программы на действия, предпринимаемые пользователем. Скорость реакции определяется временем, которое затрачивает ЭВМ на кодирование нового изображения в ответ на команду пользователя, а также длительностью передачи этой информации на экран дисплея. Во многих приложениях быстродействие играет первостепенную роль. Допустим, например, что в авиационном тренажере машина, на которой моделируется полет самолетов, могла бы реагировать на перемещения органов управления лишь спустя несколько секунд. Тогда картина на экране отображала бы

обстановку с запозданием и менялась бы неестественными скачками. В этих условиях проходящий тренировку летчик не мог бы испытывать ощущения, что он действительно пилотирует самолет; более того, сам процесс управления вызывал бы у него серьезные трудности. Вообще можно утверждать, что низкая скорость реакции всегда усложняет работу с интерактивными графическими программами, и именно этим объясняется повышенное внимание к исследованиям, направленным на разработку методов повышения быстродействия систем интерактивной графики.

Как сделать так, чтобы изображения могли расширяться, сокращаться и поворачиваться!

Во многих приложениях возникает необходимость в изменении масштаба и ориентации отдельных фрагментов выведенной на экран картины. Выполнение подобных изменений, или **преобразований** изображения основано на использовании стандартного аппарата аналитической геометрии, тригонометрии и линейной алгебры. Эти математические методы позволяют вычислить координаты концов любого отрезка прямой после поворота изображения или смены масштаба. Таким образом, применив соответствующий алгоритм, сравнительно несложно построить отрезок, получаемый в результате выполнения определенного преобразования. Проблемы возникают лишь в тех случаях, когда вычисления требуют значительных затрат времени; их можно устранить, осуществляя преобразования с помощью специального аппаратного обеспечения.

Что делать с изображениями, которые слишком велики, чтобы уместиться на экране!

Экраны дисплеев относительно невелики, и зачастую размеры картин, которые мы хотели бы воспроизвести на них, оказываются столь большими, что изображение нельзя вывести целиком. Если бы мы захотели, например, показать в увеличенном масштабе колесо, представленное на рис. 20.6, то соответствующий набор чисел уже не смог бы поместиться в буфере образа. Вероятно, в этом случае мы постарались бы вывести на тот же экран максимально большой фрагмент изображения (рис. 20.8). Процедуру, позволяющую выбирать нужные участки высвеченной на экране картины и отсекал все оставшиеся ее части, называют **кадрированием**. Кадрирование можно рассматривать как один из специальных вариантов преобразования изображений, и, действительно, оно обычно выполняется с по-

мощью тех же программных или аппаратных средств, которые реализуют другие виды преобразований,

Каким образом пользователь может вычертить на экране желаемое изображение?

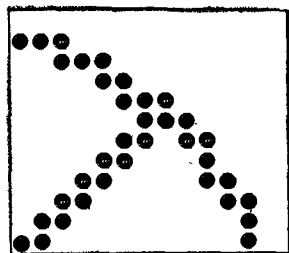


Рис. 20.8. Использование кадрирования для вывода на экран фрагмента увеличенного изображения колеса.

Возможность создавать изображения непосредственно на экране дисплея является для пользователя одним из наиболее привлекательных элементов интерактивной машинной графики. Для того чтобы сделать данный вид взаимодействия с ЭВМ максимально доступным и удобным, был разработан целый ряд различных устройств ввода графической информации, в том числе световое перо, графический планшет, «мышь» и т. д.; некоторые из них читатель уже мог увидеть на рис. 20.3 и 20.4. Когда пользователь работает с одним из этих устройств, у него возникает иллюзия, что он рисует прямо на экране дисплея. В действительности же ЭВМ отслеживает все перемещения, происходящие в устройстве ввода, и соответствующим образом корректирует выводимое изображение. Именно высокое быстродействие ЭВМ, позволяющее почти мгновенно изменять вид картинку, способствует созданию ощущения, что изображение строится пользователем непосредственно на экране.

20.5. НОВЫЕ ТИПЫ ДИСПЛЕЕВ

Электронно-лучевая трубка была и остается в наше время основным элементом устройств интерактивной машинной графики. В течение многих лет не предлагалось ничего, что могло бы стать достойной заменой ЭЛТ. Теперь можно назвать два-три таких прибора, хотя они все же во многих отношениях уступают ЭЛТ и не получили пока широкого распространения.

Почему, собственно, мы занимаемся поиском альтернативы ЭЛТ? В конце концов ЭЛТ надежна, отличается сравнительно невысокой стоимостью и способна динамично воспроизводить быстро меняющиеся изображения. Тем не менее ЭЛТ, как выходной прибор ЭВМ, имеет и свои недостатки: очень высокие уровни необходимых для ее работы напряжений, значительные габариты и вес. Устройство, которое отвечало бы нашим требованиям, должно питаться от источника с номиналом не более 10 В и быть не тяжелее и не больше по размерам, чем обычный портфель. В некоторых научно-исследовательских органи-

зациях Японии и США уже разрабатываются экспериментальные образцы дисплеев с такими параметрами. Однако все же немногие из них могут конкурировать с ЭЛТ в отношении надежности и эффективности. И ни в одном из этих приборов не устранен еще один серьезный недостаток ЭЛТ — ограниченность размеров ее экрана.

Хотя в данном разделе рассмотрены чисто технические вопросы, не имеющие, казалось бы, прямого отношения к применению ЭВМ, мы не могли обойти их, поскольку они оказывают непосредственное влияние на развитие интерактивной машинной графики. Мы без труда можем проследить, какое важное значение имеет выбор типа ЭЛТ, изучая многочисленные работы, в которых излагаются либо способы практической реализации определенных возможностей ЭЛТ, либо устранения некоторых ее недостатков. Нет сомнений, что по мере внедрения в машинную графику дисплеев новых типов посвященные им исследования будут развиваться не менее интенсивно.

20.6. УНИВЕРСАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МАШИННОЙ ГРАФИКИ

Программирование большинства видов операций ввода-вывода информации осуществляется в настоящее время стандартными способами с использованием языков высокого уровня. Так, в языках типа Паскаля предусмотрены средства для ввода и вывода файлов, а также для работы с интерактивными текстовыми терминалами. Возможность описания подобных операций в рамках распространенных языков высокого уровня не только существенно упрощает программирование, но и позволяет выполнять отработанные программы на самых разных ЭВМ. Разумеется, хотелось бы, чтобы и графические программы можно было составлять с помощью столь же простых и компактных средств. К сожалению, это далеко не всегда так.

Простоты программирования и компактности получающихся программ можно добиться путем использования **графических пакетов**, т. е. наборов стандартных подпрограмм, которые позволяли бы описывать операции доступа к графическим устройствам на языке высокого уровня. Хороший графический пакет упрощает работу программиста и дает ему возможность создавать программы, которые могут выполняться на различных ЭВМ, оснащенных дисплеями разных типов. Благодаря этому существенно сокращаются затраты на разработку программного обеспечения для графических приложений. Большинство подобных пакетов являются **универсальными** и позволяют составлять программы для решения самых разнообразных прикладных задач.

Создание универсальных графических пакетов относится к числу центральных проблем машинной графики. Поскольку пакет общего назначения должен обеспечивать выполнение широкого диапазона функций, при его разработке приходится обращаться практически ко всем разделам машинной графики. Особое значение имеет при этом организация работы с конкретными типами графических дисплеев с учетом всех их технических характеристик. Следует отметить, что последнее представляет наибольшую сложность. Появление очередной модели дисплея всякий раз ставит новые задачи перед проектировщиками графического программного обеспечения. Большие различия в конструкциях дисплеев затрудняют унификацию прикладных программ. Некоторые новейшие типы дисплеев, в том числе оснащенные буферами образа, только начинают внедряться в эксплуатацию, и для них универсальные программные средства пока отсутствуют. Это одна из серьезных проблем, ожидающих своего решения.

20.7. ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ

Любая интерактивная графическая программа требует определенной тренировки, прежде чем пользователь сможет уверенно с ней обращаться. Очень немногие программы столь же просты в работе, как описанная выше телеигра в настольный теннис. Как правило, на обучение и практику приходится затрачивать от нескольких часов до недель. В течение указанного срока пользователь получает представление о функциях, реализуемых программой, знакомится с командами, вызывающими выполнение этих функций, и учится распознавать графические образы, в виде которых оформляются результаты вычислений, выполняемых программой. Все перечисленные элементы образуют то, что принято называть **интерфейсом пользователя**; они являются составными частями программ, с помощью которых пользователь может взаимодействовать с ЭВМ и управлять ее работой. Программы, оснащенные хорошим интерфейсом, не только быстрее осваиваются пользователями, но и оказываются более простыми и эффективными в практической работе. Напротив, плохой интерфейс может вызвать такие трудности, что пользователь будет просто не в состоянии применять программу. Обращение с подобной программой сродни попытке решить головоломку из числа тех, в которых требуется прогнать несколько маленьких стальных шариков сквозь лабиринт: стараешься, не отклоняясь, идти к конечной цели, но делаешь ошибку за ошибкой и теряешь даже то небольшое, что уже достигнуто.

Когда пытаешься воспользоваться программой с плохо продуманным интерфейсом, часто не оставляет ощущение, что программист намеренно постарался, как и в случае с головоломкой, максимально затруднить действия пользователя. На самом деле такая программа лишь служит иллюстрацией того, насколько сложно организовать хороший интерфейс. Программист, разрабатывающий интерактивную графическую программу, располагает весьма ограниченным набором правил, которыми он руководствуется при проектировании интерфейса пользователя; еще меньше у него средств, с помощью которых он может проанализировать свой проект и судить о его эффективности. Создание интерфейсов пользователя, как ни одного другого элемента машинной графики, продолжает оставаться скорее предметом искусства, чем науки.

20.8. ПОСТРОЕНИЕ ИЗОБРАЖЕНИЙ ТВЕРДЫХ ТЕЛ

Посмотрите внимательнее на предмет, показанный на рис. 20.9. Кажется, что это фужер для шампанского, стоящий на поверхности, раскрашенной наподобие шахматной доски. Однако это всего лишь сгенерированное на ЭВМ изображение никогда не существовавшего объекта, точнее существовавшего лишь в форме математической модели, записанной в памяти ЭВМ. Превращение этой модели в картину, имеющую столь реальный вид, стало возможным лишь благодаря применению весьма изощренных вычислительных методов.

Задачи, связанные с построением на ЭВМ изображений, подобных представленному на рис. 20.9, можно разделить на три группы.

1. Требуется построить модели изогнутых поверхностей некоторого объекта; это делается путем разбиения поверхностей на небольшие **фрагменты**, а затем описания каждого из них с помощью параметрических уравнений, позволяющих легко преобразовать его форму.

2. Требуется выделить невидимые наблюдателю элементы объекта, которые либо находятся с обратной по отношению к наблюдателю стороны, либо закрыты другими частями объекта; данная задача, именуемая **проблемой скрытых поверхностей**, в машинной графике считается одной из классических.

3. Требуется определить, как должны быть **затонированы** видимые поверхности объекта.

Методика, использованная для тонирования изображения на рис. 20.9, была предложена Буй Туонг Фонгом, работавшим в Университете шт. Юта. Она позволяет построить машинную модель освещенного объекта, которая воспроизводит его в том виде, в каком он воспринимается глазом человека,

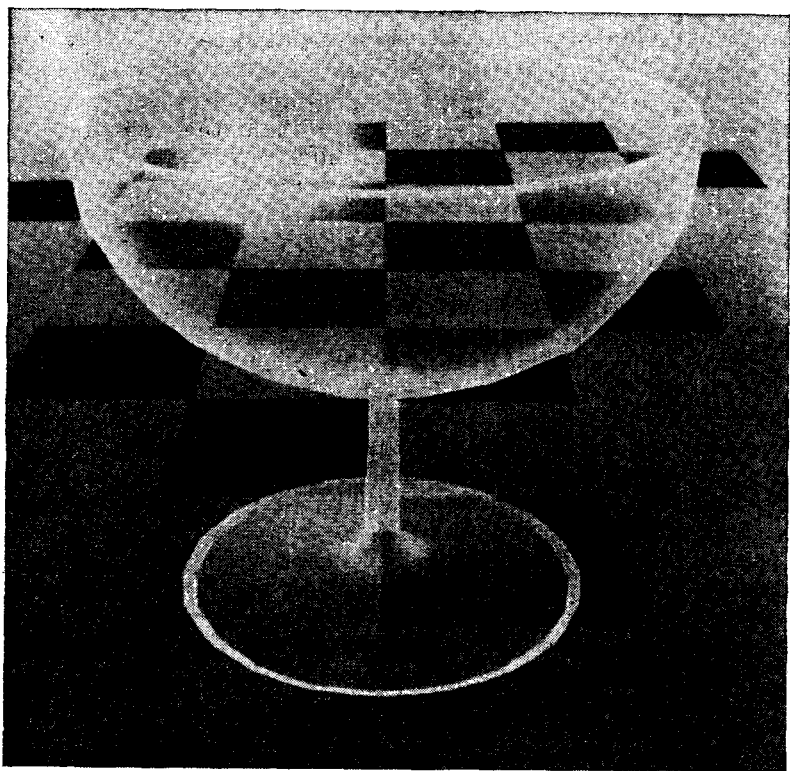


Рис. 20.9. Машинное изображение смоделированной трехмерной композиции (с разрешения Университета шт. Юта).

Построение изображений твердых тел является наиболее разработанным в математическом отношении разделом машинной графики. Это направление стало темой ряда оригинальных научных исследований. Их результаты были применены для решения многих прикладных задач и, пожалуй, наиболее успешно для моделирования полетов, о чем шла речь в приведенном ранее примере. Методы воспроизведения твердых тел имеют много потенциальных приложений в различных областях: авиа- и автомобилестроении, архитектуре, городском планировании и т. д. Условием широкого распространения этих методов в указанных областях является создание более быстродействующих и менее дорогостоящих средств отображения. Это остается одной из наиболее актуальных проблем интерактивной машинной графики.

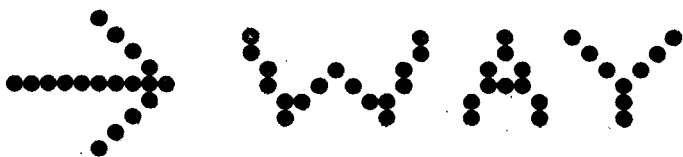


Рис. 20.10. Линии и буквы на экране растрового дисплея, увеличенные для того, чтобы показать отдельные пикселы.

20.9. ТЕХНИКА РАСТРОВЫХ ИЗОБРАЖЕНИЙ

Описанный выше дисплей с буфером образа относится к устройствам **растрового типа**. Наименьшим элементом изображения, который может быть показан на экране такого дисплея, является маленькое светящееся пятно, именуемое по-разному — либо **точкой**, либо **пикселем**. Чтобы изображение в растровом дисплее могло восприниматься глазом, оно должно быть составлено из многих сотен пикселей, для генерации каждого из которых требуется отдельная машинная команда. Прямые и кривые линии образуются из сгруппированных наборов пикселей. Для вывода текстового символа, т. е. буквы или цифры, формируется определенная совокупность пикселей, которую называют **матрицей**. На рис. 20.10 приведены увеличенные изображения отрезков прямых и букв, полученные на растровом дисплее.

Самые первые графические дисплеи были растрового типа. Буферы образа тогда еще не существовали, и для построения изображения приходилось последовательно пересылать из ЭВМ координаты всех его точек. Разумеется, при таком способе общее число точек, которое могло быть выведено без мерцания, было весьма ограниченным.

На смену этим несовершенным растровым дисплеям в середине 60-х годов пришли дисплеи векторного типа. Подобные устройства позволяют вычерчивать сразу целые отрезки прямых линий без вывода образующих их точек по отдельности; благодаря этому на них может быть получена значительно более высокая плотность элементов изображения, чем на растровых дисплеях. Кроме того, при использовании векторных дисплеев отпадает необходимость в вычислении координат каждой точки.

Несмотря на то что первые модели растровых дисплеев давно вышли из употребления, созданная в тот период техника программирования сохраняет свое значение и теперь. Разработанные для них методы поточечного вывода изображений нашли применение при создании программного обеспечения для дисплеев с буферами образа, которые вновь требуют вычислять координаты всех точек по отдельности. Для нас же эти методы представляют интерес еще и потому, что они служат хорошей иллюстрацией **метода приращений**, столь часто применяемого в машинной графике.

20.10. СИСТЕМЫ КООРДИНАТ

В растровой технике повсюду используются прямоугольные **декартовы системы координат**. Адреса точек задаются их координатами по осям x и y ; значения x возрастают слева направо, y — снизу вверх (рис. 20.11).

Точки высвечиваются на экране в соответствии с цифровыми сигналами, вырабатываемыми ЭВМ. Это означает, что положения точек могут быть заданы с произвольно высокой точностью, поскольку она ограничена только диапазоном изменения дискретных величин, выдаваемых машиной. Например, если координаты x и y поступают в виде 10-разрядных двоичных чисел, существует 1024 (2^{10}) различных значений x и 1024 значений y . Таким образом, весь экран можно представить как массив из 1024×1024 позиций, в любой из которых может быть высвечена точка.

Что же обуславливает точность воспроизведения изображений на дисплее? В большинстве случаев точность ограничивается **разрешающей способностью** его экрана. Этот показатель определяется как максимальное число различаемых глазом отдельных точек, которые можно разместить внутри некоторой области экрана. Разрешающая способность типового дисплея составляет примерно 100 точек на дюйм (2,54 см). Это значит, что глаз может различить две точки, расположенные друг от друга на расстоянии $1/100$ дюйма. Значительное увеличение точности задания координат по сравнению с разрешающей способностью экрана не сулит никакого выигрыша, поскольку наблюдатель все равно не сумеет заметить разницу. Если, напротив, точность существенно меньше разрешающей способности, то на экране будут высвечиваться отчетливо различимые точки, между которыми уже нельзя вставить других; при этом в линиях будут наблюдаться разрывы. Отсюда ясно, что при проектировании дисплея точность определения координат следует устанавливать близкой к разрешающей способности его экрана.

Задавшись некоторой точностью координат и размерами экрана, нетрудно определить количество адресуемых точек. Создание дисплея, имеющего квадратный экран со стороной более 12 дюймов (около 30 см) при разрешении 100 точек на дюйм является достаточно сложной инженерной задачей. Поэтому большинство дисплеев обеспечивает адресацию не более 1200 позиций по каждой координате. Типовым является число 1024, так как оно соответствует полному диапазону изменения целой десятиразрядной двоичной переменной, задающей координату точки. Однако встречаются конструкции и со значительно большим растром (до 4096×4096 адресуемых точек).

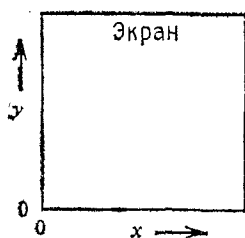


Рис. 20.11. Декартова система координат на экране дисплея.

и с гораздо более скромным (всего 256×256).

Подводя итог, повторим, что выпускаемые в настоящее время дисплеи для интерактивных графических систем работают в декартовой системе координат с 10-разрядной адресацией по каждой координате; экраны дисплеев, как правило, бывают квадратными со стороной около 10 дюймов (~ 25 см). Обычной является практика использования целых чисел для задания координат; начало координат чаще

всего помещается в нижнем левом углу экрана, как показано на рис. 20.11.

20.11. МЕТОД ПРИРАЩЕНИЙ

Человек, приехавший в незнакомый город, разыскивая нужное ему место, часто интуитивно пользуется методом приращений. Если он очутился у дома № 203 по Мэйн-стрит, а ему необходимо попасть в дом № 735, он доберется туда, отыскав предварительно любой дом с номером, большим 203, например № 205. Найдя такой, он продолжает идти в том же направлении, проходит мимо 207-го и т. д., пока не окажется у нужного ему дома № 735. Использование номеров, присваиваемых домам в порядке их расположения вдоль улиц, значительно облегчает поиск. Совсем не так обстоит дело в Токио, где дома нумеруются согласно времени их постройки. Здесь обычный метод приращений уже неприменим. Широкое распространение метода приращений в машинной графике объясняется также тем, что с его помощью удастся существенно упростить многие операции. Несколько позже будет показано, как благодаря использованию этого метода можно более эффективно реализовывать операции построения поверхностей и тонирования генерируемых на ЭВМ изображений твердых тел. Начнем же мы изучение метода приращений в настоящем разделе, поскольку основанные на нем алгоритмы применяются для вычерчивания линий в растровых дисплеях.

Вычисления в приращениях являются одной из разновидностей метода итераций, в котором каждая итерация выполняется очень просто, но за счет дополнительного расхода памяти для хранения **состояния**, т. е. некоторой информации, позволяющей контролировать ход вычислений. Так, проложему, отыскивающему дом № 735, необходимы лишь три элемента, описывающие текущее состояние: направление, в котором он движется, номер дома, мимо которого он проходит в данный момент, и номер,

нужного ему дома. Если он дойдет до дома с номером, лежащим вне промежутка между двумя последними номерами, то, согласно алгоритму, прохожий должен начать двигаться в противоположном направлении. Этот алгоритм, реализованный в виде приведенной ниже программы на языке Паскаль, является простейшим примером использования метода приращений.

```
procedure  найти_дом (нужный_дом: integer);  
  var      текущий_дом, t, направление: integer;  
begin  
  направление := 1;  
  текущий_дом := считанный_номер_дома;  
  while    текущий_дом < > нужный_дом do begin  
    перейти_к_следующему_дому (направление);  
    t := считанный_номер_дома;  
    if (t > Max (текущий_дом, нужный_дом)) or  
       (t < Min (текущий_дом, нужный_дом)) then  
      направление := - направление;  
    текущий_дом := t  
  end;  
end;
```

Характерной особенностью метода приращений является то, что для получения конечного результата предварительно необходимо определить все промежуточные. Например, когда нужно построить по приращениям отрезок прямой, мы, задавшись координатами одного из его концов, последовательно вычисляем координаты лежащих на нем точек, пока не дойдем до противоположного конца. Таким образом, одна итерационная процедура порождает целую совокупность полезных результатов.

20.12. АЛГОРИТМЫ ВЫЧЕРЧИВАНИЯ ОТРЕЗКОВ

Отрезки прямых линий очень часто встречаются на изображениях, получаемых с помощью средств машинной графики. Они являются одними из наиболее распространенных элементов структурных схем, гистограмм и графов, архитектурных и машиностроительных чертежей, логических схем, градостроительных планов и т. д. Кроме того, наборы коротких отрезков прямых могут служить достаточно хорошей аппроксимацией кривых линий. Учитывая, что отрезки прямых имеют такое значение в машинной графике, стоит задуматься над тем, как повысить качество их воспроизведения,

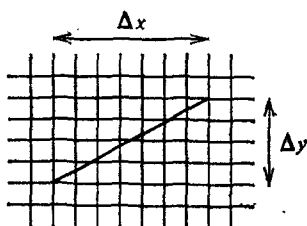


Рис. 20.12. Отрезок прямой, соединяющий два узла сетки, может не проходить через другие узлы.

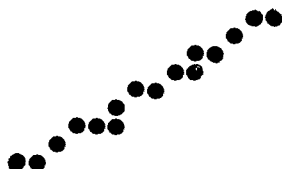


Рис. 20.13. Результат работы плохого алгоритма для вычерчивания прямых.

Каким же требованиям должен отвечать хороший алгоритм, предназначенный для вычерчивания отрезков? Прежде чем ответить на этот вопрос, нам придется выяснить, какими навыками следует овладеть чертежнику, чтобы стать квалифицированным специалистом. Он должен научиться проводить линии совершенно прямо и так, чтобы они начинались и заканчивались в строго определенных точках. Каждый отрезок прямой на всем его протяжении должен иметь постоянную черноту, или **плотность**, а в точках сопряжения отрезков их плотности должны совпадать. И наконец, приобретая определенный опыт, чертежник выполняет все построения быстро и уверенно.

Совершенно аналогичные требования предъявляются и к линиям, вычерчиваемым с помощью ЭВМ:

Линии должны выглядеть прямыми. Растровая техника идеально приспособлена для генерации линий, параллельных координатным осям или расположенных к ним под углом в 45° . Во всех остальных случаях возникает проблема — ведь даже если отрезок прямой начинается и кончается в адресуемых точках, то в промежутке на ней может не оказаться ни одной такой точки. Подобная линия показана на рис. 20.12. В этих случаях линию приходится аппроксимировать, подбирая ближайшие к ней точки раstra. Если они выбраны правильно, линия выглядит достаточно прямой; если нет, то получаются ломаные типа той, что приведена на рис. 20.13.

Линии должны заканчиваться в определенных точках. Даже если прямая воспроизведена хорошо, концы ее могут оказаться несколько смещенными. Этот эффект проявляется в виде небольших зазоров между конечной точкой одного отрезка и начальной точкой другого, а также приводит к накоплению суммарной ошибки (рис. 20.14).

Линии должны иметь постоянную плотность. Если светлая линия вычерчивается на темном фоне, для наблюдателя плотность определяется ее яркостью, если темная линия лежит на

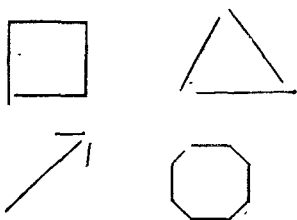


Рис. 20.14. Фигуры, составленные из отрезков с плохо сопрягающимися концами.

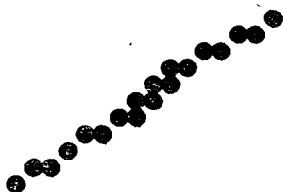


Рис. 20.15. Неравномерная плотность линии, вызванная сгущением точек.

светлом фоне — степенью ее черноты. В обоих случаях плотность линии пропорциональна количеству участвующих в построении точек, отнесенному к ее длине. Для того чтобы плотность была постоянна, расстояния между точками должны быть равными. Однако таким свойством могут обладать только линии, либо параллельные осям координат, либо наклоненные к ним под углом в 45° . В остальных случаях следует добиваться как можно более равномерного распределения точек. Иначе всякое их сгущение будет восприниматься в виде затемненных или, наоборот, высветленных участков прямой (см. пример, приведенный на рис. 20.15).

Плотность линий не должна зависеть от их длины и наклона. Данное условие выполнить весьма не просто. Как мы уже выяснили, чтобы получить равномерную плотность линии, число точек, приходящихся на единицу ее длины, должно быть постоянным. Следовательно, прежде чем вывести линию на экран, необходимо определить ее точную длину, что связано с операцией вычисления квадратного корня. Кроме того, нужно иметь возможность по мере вычерчивания линии регулировать частоту вывода точек. И то и другое достаточно сложно. Все, что обычно удается сделать — это вычислить приближенную оценку длины и осуществить построение, воспользовавшись алгоритмом, обеспечивающим постоянство плотности в пределах, соответствующих погрешности этой оценки.

Линии должны вычерчиваться с большой скоростью. Желательно, чтобы при работе в интерактивном режиме линии появлялись на экране максимально быстро. Это означает, что используемый для их построения алгоритм должен содержать как можно меньше вычислительных операций; однако идеальным решением является реализация всех вычислительных процедур с помощью специализированных аппаратных средств.

Все обсуждаемые далее алгоритмы работают в приращениях, ввиду чего им присущи некоторые общие особенности. В част-

ности, каждый из них генерирует две основные последовательности чисел; это приращения вертикальных и горизонтальных координат точек, отображающих линию. При построении вертикальных линий выдаются только значения y , в то время как для линии с углом наклона 45° значения x и y с постоянным шагом. Для любой конкретной линии в каждом из приведенных ниже алгоритмов вычисляется одно и то же количество значений x и y , поскольку их суммы должны быть равны величинам Δx и Δy , т. е. смещениям конечной точки отрезка относительно начальной по соответствующим координатам (см. рис. 20.12). Алгоритмы отличаются только порядком выдачи результатов и процедурами, применяемыми для их вычисления.

Симметричный ЦДА

Цифровые дифференциальные анализаторы (ЦДА) осуществляют построение прямых, заданных дифференциальными уравнениями. Далее в этой главе будет показано, что ЦДА могут применяться и для вычерчивания кривых, если последние могут быть описаны обыкновенными дифференциальными уравнениями. Уравнение прямой имеет особенно простой вид:

$$dy/dx = \Delta y/\Delta x. \quad (20.1)$$

Столь же просто устроен и ЦДА, предназначенный для построения прямых.

Принцип действия ЦДА состоит в следующем. Переменным x и y придаются небольшие приращения, пропорциональные соответствующим первым производным. В случае прямой эти производные постоянны и пропорциональны Δx и Δy . Если бы дисплей обладал абсолютной точностью, мы могли бы строить прямые, прибавляя к x и y на каждом шаге величины $\epsilon \Delta x$ и $\epsilon \Delta y$, где ϵ — некоторая произвольная малая константа (рис. 20.16).

В реальных условиях точность дисплеев ограничена, и мы вынуждены использовать лишь адресуемые точки. Для этого после вычисления очередной пары приращений производится их округление до ближайших целых чисел; результирующие значения x и y служат координатами точки, выводимой на экран.

Вместо округления иногда используется эффект переполнения при выполнении операции сложения. В этом случае переменные x и y хранятся в регистрах, причем их целые и дробные части содержатся по отдельности. Значения приращений, оба меньшие единицы, прибавляются на каждом шаге к дробным частям, и всякий раз, когда происходит переполнение, целая часть соответствующего числа изменяется на единицу. Далее

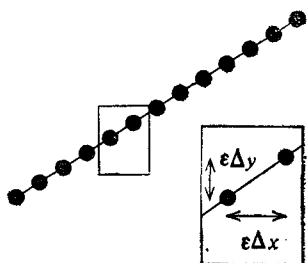


Рис. 20.16. Построение прямой методом приращений в отсутствие погрешностей.

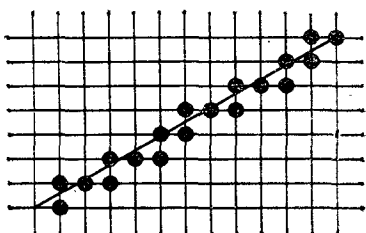


Рис. 20.17. Прямая, полученная с помощью симметричного ЦДА.

целые части x и y используются в качестве координат точек, из которых формируется прямая. При таком способе вычислений вместо округления происходит **усечение результата** и, для того чтобы компенсировать возникающее постоянное смещение, на первом шаге к содержимому обоих регистров ЦДА добавляется число 0,5.

Одно из преимуществ описанного метода состоит в том, что он позволяет контролировать изменения переменных x и y , а также следить за тем, чтобы одни и те же точки не высвечивались повторно. Встроенный в ЦДА индикатор переполнения выдает сигналы, по которым выводятся очередные точки, воспроизводящие отрезок прямой. Отметим, что погрешность представления приращений, а также дробных частей, содержащихся в регистрах, не должна превосходить шага координатной сетки экрана дисплея; иначе при построении отрезков большой длины будут накапливаться значительные ошибки.

Вид линий, генерируемых с помощью ЦДА, зависит от выбора коэффициента ϵ . Для **симметричных ЦДА** он задается равным 2^{-n} , причем.

$$2^{n-1} \leq \max(|\Delta x|, |\Delta y|) < 2^n. \quad (20.2)$$

На практике величина ϵ устанавливается как обратная по отношению к полученной в ЦДА оценке длины отрезка, т. е. 2^n . Прямая, построенная с использованием симметричного ЦДА, изображена на рис. 20.17; работа симметричного ЦДА иллюстрируется схемой, приведенной на рис. 20.18.

Симметричный ЦДА обеспечивает высокую точность воспроизведения прямых, поскольку смещение выводимой на экран точки относительно ее вычисленного положения никогда не превышает половины шага координатной сетки. Логика работы симметричного ЦДА чрезвычайно проста; использование в качестве коэффициента ϵ отрицательной степени числа 2

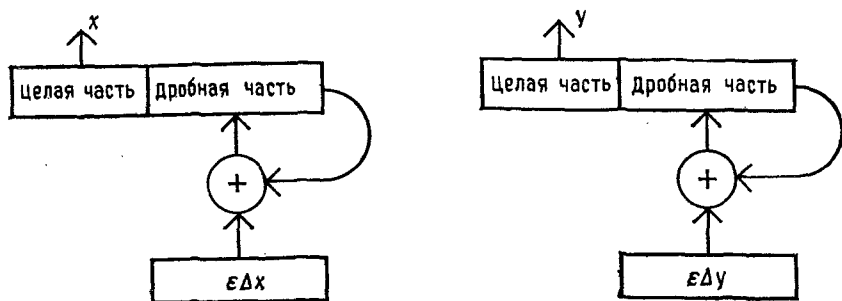


Рис. 20.18. Принцип действия симметричного ЦДА.

позволяет исключить операцию деления и вычислять приращение, просто сдвигая Δx и Δy на соответствующее число разрядов перед суммированием с содержимым регистров. Таким образом, один шаг построения прямой требует выполнения всего двух операций сложения.

Простой ЦДА

В симметричном ЦДА с целью сокращения числа операций оценка длины отрезка принимается равной степени 2. В то же время сам принцип действия ЦДА не накладывает ограничений на выбор указанной оценки и соответствующего ей значения ϵ ; необходимо лишь, чтобы ни $\epsilon\Delta x$, ни $\epsilon\Delta y$ не превосходили по величине единицы.

В простом ЦДА в качестве оценки длины отрезка принимается максимальное из значений Δx и Δy , в силу чего либо $\epsilon\Delta x$, либо $\epsilon\Delta y$ оказываются равными единице. Данное обстоятельство позволяет заменить один из регистров ЦДА обыкновенным счетчиком. В результате простой ЦДА на каждом шаге построения отрезка всегда выдает единичные приращения по более быстро меняющейся координате (см. пример на рис. 20.19). Ниже приведена программа на языке Паскаль, реализующая алгоритм работы простого ЦДА:

```

procedure ЦДА (x1, y1, x2, y2: integer);
  var длина, i: integer; x, y, приращение_x, приращение_y: real;
begin
    длина := abs (x2 - x1);
    if abs (y2 - y1) > длина then длина := abs (y2 - y1);
    приращение_x := (x2 - x1)/длина;
    приращение_y := (y2 - y1)/длина;
    x := x1 + 0,5; y := y1 + 0,5;

```

```

for  $i := 1$  to длина do
  begin
    вывести (целая часть ( $x$ ), целая часть ( $y$ ));
     $x := x + \text{приращение}_x$ ;
     $y := y + \text{приращение}_y$ ;
  end;
end;
```

Простой ЦДА обеспечивает не меньшую точность построения, чем симметричный, но генерирует иную последовательность точек, поскольку в нем принят другой способ оценки длины отрезка. Логика его работы в основном проще, за исключением того, что в самом начале для определения коэффициента приращения приходится однократно выполнять операцию деления. Простой ЦДА является идеальной основой для построения программных генераторов векторов. В то же время необходимость введения дополнительной логики для выполнения деления несколько усложняет его аппаратную реализацию.

Алгоритм Брезенхэма

Интересный алгоритм для вычерчивания прямых был предложен Брезенхэмом. Подобно простому ЦДА, на каждом шаге вычислений он изменяет значение одной из координат на ± 1 . Значение второй координаты либо также меняется, либо остается прежним, что зависит от заданной величины погрешности, обеспечиваемой алгоритмом. Эта погрешность определяется как расстояние выведенной на экран точки от идеальной прямой, замеренное в направлении, перпендикулярном более быстро меняющейся координате. В примере, показанном на рис. 20.20, скорость изменения координат точки вдоль оси x выше, ввиду чего ошибка, обозначенная как e , замерена вдоль оси y . Приведенное ниже описание алгоритма предполагает именно такую ориентацию прямой.

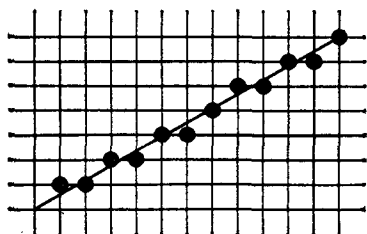


Рис. 20.19. Прямая, полученная с помощью простого ЦДА.

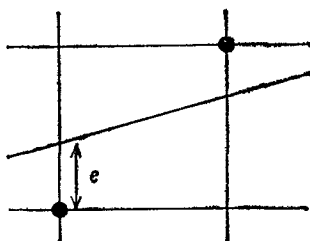


Рис. 20.20. Графическая иллюстрация алгоритма Брезенхэма.

На каждом шаге алгоритма коэффициент наклона прямой $\Delta y/\Delta x$ суммируется с ошибкой e . Предварительно анализируется знак e , по которому определяется, должна ли измениться координата y текущей точки. Положительное значение e указывает, что истинная прямая лежит выше текущей точки; в этом случае y получает приращение, а из e вычитается 1. Если ошибка e отрицательна, значение координаты y остается неизменным. В указанной форме данный алгоритм может быть реализован в виде следующей программы на языке Паскаль:

```
{Примечание: переменные  $x$ ,  $y$ ,  $\delta x$ ,
 $\delta y$  — типа integer;  $e$  — типа real}
 $e := (\delta y / \delta x) - 0,5$ ;
for  $i := 1$  to  $\delta x$  do begin
    вывести ( $x$ ,  $y$ );
    if  $e > 0$  then begin
         $y := y + 1$ ;
         $e := e - 1$ 
    end
     $x := x + 1$ ;
     $e := e + (\delta y / \delta x)$ 
end;
```

Явным недостатком приведенного варианта алгоритма является необходимость выполнения операций деления для вычисления начального значения и приращения e . Однако эти операции можно исключить, поскольку логика алгоритма не изменится, если умножить e на некоторую константу — ведь анализируется только знак приращения. Действительно, умножив e на $2\Delta x$, получаем новый вариант программы, в котором нет ни делений, ни умножений:

```
{Примечание: все переменные — типа integer}
 $e := 2 * \delta y - \delta x$ ;
for  $i := 1$  to  $\delta x$  do begin
    вывести ( $x$ ,  $y$ );
    if  $e > 0$  then begin
         $y := y + 1$ ;
         $e := e + (2 * \delta y - 2 * \delta x)$ ;
    end
    else  $e := e + 2 * \delta y$ ;
     $x := x + 1$ 
end;
```

Для реализации алгоритма Брезенхэма в законченном виде необходимо предусмотреть анализ всех вариантов, кроме уже рассмотренного, при котором выполняется условие $0 \leq \Delta y \leq \Delta x$.

В то же время алгоритм может быть несколько упрощен за счет использования только целых переменных. Аналогично простому ЦДА алгоритм Брезенхэма гарантирует отсутствие повторной выдачи одних и тех же точек. Кроме того, он не требует выполнения операций умножения и деления, что позволяет очень просто реализовать алгоритм с помощью аппаратных средств или простейших микропроцессоров.

20.13. ГЕНЕРАТОРЫ ОКРУЖНОСТЕЙ

В некоторых приложениях, особенно тех, что связаны с разработкой машиностроительных чертежей, очень часто приходится воспроизводить на экране окружности или дуги окружностей. Для построения этих фигур был предложен ряд алгоритмов, основанных на методе приращений. Такие алгоритмы имеют большое значение, поскольку в большинстве дисплеев, даже оснащенных схемными генераторами векторов, не предусмотрено никаких средств для вычерчивания окружностей. В тех случаях, когда имеются аппаратные средства для построения прямых, генераторы окружностей, работающие в приращениях, могут применяться для определения точек сопряжения коротких сегментов; если таких средств нет, то генераторы окружностей можно использовать для вычисления координат близко расположенных друг к другу точек, предназначенных для воспроизведения на растровых дисплеях. Ниже мы обсудим один из алгоритмов генерации окружностей, представляющий специальный вариант ЦДА.

ЦДА для генерации окружностей

Как уже говорилось, принцип работы ЦДА допускает построение не только прямых, но и кривых линий; одной из них является дуга окружности. Запишем дифференциальное уравнение окружности с центром в начале координат:

$$dy/dx = -x/y. \quad (20.3)$$

Нетрудно видеть, что один из вариантов ЦДА для построения такой окружности можно реализовать, используя в качестве приращений величины $(-ex)$ и ey :

$$\begin{aligned} x_{n+1} &= x_n + ey_n, \\ y_{n+1} &= y_n - ex_n. \end{aligned} \quad (20.4)$$

Согласно этим соотношениям, значения приращений должны на каждом шаге определяться заново, однако все вычисления можно свести всего к паре операций сдвига и одной операции формирования дополнительного кода, если коэффициент e

положить равным отрицательной степени числа 2; для того чтобы координаты рядом расположенных точек отличались не более, чем на один шаг сетки, данный коэффициент должен быть равен 2^{-n} при условии, что

$$2^{n-1} \leq r \leq 2^n, \quad (20.5)$$

где r — радиус окружности.

К сожалению, приведенный вариант алгоритма фактически выдает не дугу окружности, а отрезок спирали. Каждый шаг делается в направлении, перпендикулярном текущему радиусу окружности, вследствие чего очередная точка оказывается несколько дальше от ее центра, чем предыдущая. Эту проблему удается, однако, без труда разрешить, воспользовавшись для вычисления y_{n+1} не x_n , а x_{n+1} :

$$\begin{aligned} x_{n+1} &= x_n + ey_n, \\ y_{n+1} &= y_n - ex_{n+1}. \end{aligned} \quad (20.6)$$

В основе такой модификации лежит следующий подход. Уравнения (20.4) можно представить в векторно-матричной форме:

$$\begin{bmatrix} x_{n+1} & y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n & y_n \end{bmatrix} \begin{bmatrix} 1 & -e \\ e & 1 \end{bmatrix}. \quad (20.7)$$

Определитель матрицы в (20.7) равен $1 + e^2$, а не 1; из-за этого вычерчиваемая кривая и принимает вид расходящейся спирали. Если бы определитель был в точности равен единице, кривая замкнулась. Мы можем этого достигнуть, изменив лишь один элемент матрицы:

$$\begin{bmatrix} x_{n+1} & y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n & y_n \end{bmatrix} \begin{bmatrix} 1 & -e \\ e & 1 - e^2 \end{bmatrix}. \quad (20.8)$$

После несложных преобразований данное соотношение переходит в уравнения (20.6).

Центры окружностей, которые строятся с помощью ЦДА, не обязательно должны помещаться в начале координат, как это предполагалось при выводе уравнений (20.6). Последние распространяются и на случай произвольного расположения центра, если в них для вычисления новых значений (x_{n+1}, y_{n+1}) использовать в качестве (x_n, y_n) смещения текущей точки относительно центра по соответствующим координатам. Данный алгоритм весьма удобен для аппаратной реализации.

Вероятно, некоторые читатели уже обнаружили, что уравнения (20.6) описывают движение точки не по окружности, а по эллипсу. Если коэффициент e выбран достаточно большим, то

получающиеся кривые выглядят заметно сплюснутыми. При уменьшении ε этот эффект практически исчезает, однако в то же время существенно возрастает объем вычислений.

Не вызывает принципиальных трудностей и создание ЦДА, формирующего правильные окружности; реализуемый им алгоритм описывается уравнениями

$$\begin{aligned}x_{n+1} &= x_n \cos \theta + y_n \sin \theta, \\ y_{n+1} &= y_n \cos \theta - x_n \sin \theta.\end{aligned}\tag{20.9}$$

Поскольку угол θ обычно выбирается малым, значения функций $\cos \theta$ и $\sin \theta$ вычисляются сравнительно просто, причем они постоянны для всех окружностей определенного радиуса. Таким образом, целесообразно использовать именно эти уравнения, если, конечно, выполнение операций умножения не влечет чрезмерных затрат. Данный подход был развит Кохеном, который обобщил его на случай конических сечений общего вида.

Были предложены также модернизированные варианты еще нескольких алгоритмов генерации окружностей, позволяющие применять их и для построения более широкого класса кривых.

20.14. ВЕКТОРНЫЕ ДИСПЛЕИ

Изображения, формируемые с помощью ЭВМ, можно разделить на два класса — контурные и тоновые. Примеры изображений каждого типа показаны на рис. 20.21; ранее были приведены и другие примеры. Указанные два класса изображений не только отличаются внешне, но для их построения применяются совершенно разные методы. Создание контурных рисунков во многих отношениях проще, поскольку алгоритмы их генерации не столь громоздки, требуют меньшего объема информации и могут быть реализованы с помощью более доступного оборудования (во всяком случае, так обстояло дело до недавнего времени). Собственно говоря, тоновые изображения вообще невозможно было воспроизвести на экране, пока в конце 60-х годов не появились дисплеи, оснащенные буферами образа. Что же касается алгоритмов для генерации таких изображений, то они пока находятся в экспериментальной стадии. Поэтому, учитывая, что проблемы построения контуров изображений исследованы значительно глубже, на них и было сосредоточено наше внимание в первой половине настоящей главы.

Разумеется, дисплей с буфером образа позволяют выводить на экран отрезки прямых. Используя один из рассмотренных выше алгоритмов на основе метода приращений, можно определять пиксели, через которые проходит требуемый отрезок,

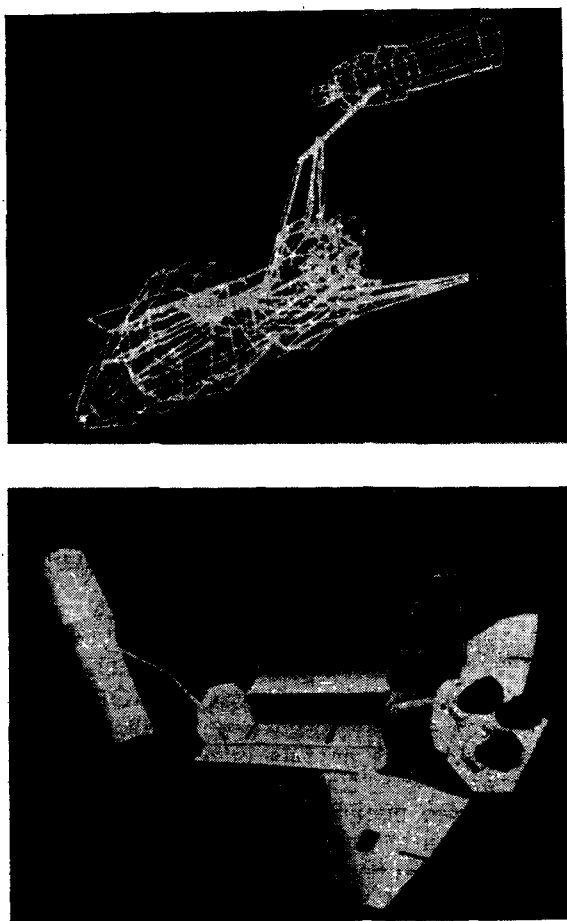


Рис. 20.21. Полученные с помощью ЭВМ изображения американского космического корабля многоразового использования «Спейс шаттл».

и соответствующим образом изменять содержимое ячеек памяти, образующих буфер образа. Это один из наиболее распространенных на практике способов построения контурных изображений; в частности, все приведенные в этой главе рисунки такого типа получены на интерактивных дисплеях с буфером образа. Впрочем, применение буфера образа имеет и свои недостатки: так, он не позволяет добиться высокого качества при вычерчивании линий малой кривизны, так как на экране почти всегда заметны изломы, обусловленные эффектами квантова-

ния; кроме того, его быстроедействие иногда оказывается недостаточным для выполнения быстрых преобразований изображений в интерактивном режиме. Если указанные проблемы являются существенными, целесообразно использовать векторные дисплеи, специально предназначенные для воспроизведения отрезков прямых линий.

20.15. ДИСПЛЕИ И ДИСПЛЕЙНЫЕ КОНТРОЛЛЕРЫ

Дальнейшее наше обсуждение мы посвятим принципам построения основных устройств машинной графики — дисплеев и дисплейных контроллеров. Назначением дисплея является преобразование электрических сигналов в видимые изображения. Дисплейный контроллер осуществляет связь между ЭВМ и дисплеем, принимая информацию от ЭВМ и формируя на ее основе сигналы, необходимые для работы дисплея. Контроллер реализует ряд функций, в том числе согласование уровней напряжений на выходе ЭВМ и входе дисплея, буферизацию сообщений с целью компенсации различий в скорости выполнения операций, генерацию сигналов для воспроизведения отрезков прямых и алфавитно-цифровых символов.

В целом задача контроллера заключается в устранении разного рода несовместимостей характеристик дисплея и ЭВМ, а также в компенсации некоторых ограничений, присущих дисплею, благодаря чему как для самой ЭВМ, так и для ее программиста доступ к этому устройству значительно упрощается. Однако это лишь главная задача, для решения которой первоначально и предназначались контроллеры. Современные дисплейные контроллеры оснащаются дополнительной электроникой, выполняющей такие функции, как масштабирование и вращение изображений, которые ранее приходилось реализовывать с помощью программных средств; цель введения указанной аппаратуры состоит главным образом в повышении быстрогодействия графических систем.

20.16. ДИСПЛЕИ

В большинстве приложений машинной графики существенную роль играет качество воспроизведения изображений. Поэтому неудивительно, что усилия исследователей в значительной степени были направлены на разработку эффективных средств визуализации. В 50-х годах, когда эти работы только разворачивались, электронно-лучевая трубка являлась единственным прибором, с помощью которого электрические сигналы, выдаваемые ЭВМ, могли с большой скоростью преобразовываться в видимые изображения. В то время ЭЛТ были весьма

дороги, их экраны имели очень небольшой размер и светились достаточно тускло. Однако по прошествии лет технология производства ЭЛТ была значительно усовершенствована, в результате чего удалось создать целый ряд чрезвычайно эффективных приборов, применяемых в графических дисплеях различных типов. Одновременно и сами ЭЛТ оказали большое влияние на путь развития интерактивной машинной графики.

Наряду с продолжающимся совершенствованием ЭЛТ велись интенсивные исследования по созданию других приборов, способных стать их достойной заменой. Некоторые из этих работ увенчались успехом, и было предложено несколько принципиально новых способов преобразования электрических сигналов в видимые изображения. Многие из устройств нового поколения предназначались в первую очередь для использования в производстве телевизионной техники и алфавитно-цифровых индикаторов; однако большинство этих приборов находит применение и в машинной графике. Вместе с тем следует отметить, что ни один из них не смог пока занять места ЭЛТ в качестве основного графического устройства визуализации.

20.17. КОНСТРУКЦИЯ ЭЛТ

Основные элементы ЭЛТ показаны на рис. 20.22. В узкой горловине конической, герметически запаянной стеклянной колбы помещается **электронная пушка**, которая испускает остро сфокусированный пучок электронов, движущихся с высокой скоростью. Расширенный конец трубки, являющийся ее передней частью, имеет более или менее плоскую поверхность, покрытую изнутри слоем **люминофора**, который начинает светиться под воздействием бомбардирующих его электронов. Энергией пучка можно управлять, что позволяет регулировать

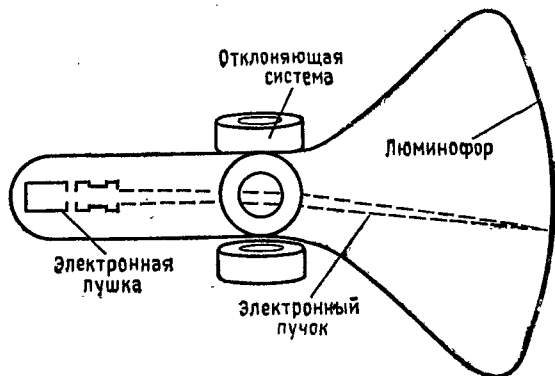


Рис. 20.22. Упрощенная конструкция ЭЛТ.

интенсивность свечения люминофора или вовсе прекращать испускание света. Снаружи трубки, в том месте, где конус переходит в горловину, смонтирована отклоняющая система, образованная набором электромагнитных катушек; токи, протекающие через эти катушки, заставляют пучок электронов отклоняться и направляют его в требуемые точки лицевой поверхности трубки. После того как пучок удаляется с некоторого участка люминофора, интенсивность его свечения быстро падает, и для того, чтобы наблюдатель воспринимал эту точку как постоянно светящуюся, необходимо периодически, причем достаточно часто, направлять на нее пучок; данный процесс, именуемый **регенерацией**, обычно осуществляется с частотой повторения не менее 30 раз в секунду.

Электронная пушка

В электронной пушке для фокусирования пучка и ускорения электронов используются свойства **электростатического поля**. Такое поле возникает, когда две поверхности заряжаются до различных потенциалов (уровней напряжения); попав в поле, электроны начинают смещаться в сторону поверхности, имеющей более высокий положительный потенциал. Сила, действующая на отдельный электрон, прямо пропорциональна разности потенциалов.

Электронная пушка предназначена для создания пучка электронов, обладающего следующими свойствами:

1. Он должен быть точно сфокусирован таким образом, чтобы в месте попадания пучка на люминофор возбуждалось точечное светящееся пятно.
2. Электроны в пучке должны иметь высокую скорость, поскольку яркость пятна возрастает с увеличением скорости.
3. Должны быть предусмотрены средства управления потоком электронов, позволяющие регулировать интенсивность свечения точки, на которую направлен пучок.

Электронная пушка, отвечающая перечисленным требованиям, содержит ряд узлов, показанных на рис. 20.23. Электроны испускаются **катодом**, который нагревается изнутри нитью накала. Катод заключен в **управляющую сетку** — металлический цилиндр с отверстием в торце, через которое вылетают электроны. На эту сетку подается потенциал более низкий, чем у катода, за счет чего создается электростатическое поле, так направляющее электроны, что их траектории на выходе из отверстия пересекаются, образуя точечный источник; это упрощает последующую фокусировку пучка. Варьируя потенциал сетки, можно изменять уровень эмиссии электронов, или **ток пучка**, и тем самым регулировать яркость изображения; можно также совсем прекратить вылет электронов.

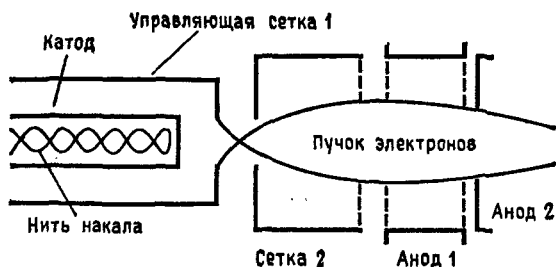


Рис. 20.23. Электронная пушка ЭЛТ.

Фокусировка пучка осуществляется с помощью **фокусирующей системы**, которая включает не менее двух металлических пластин, имеющих разные потенциалы. Они формируют тороидальное электростатическое поле, захватывающее разлетающиеся из источника электроны, и заставляющее их двигаться в направлении продольной оси системы. В результате образуется точно сфокусированный пучок, стягивающийся в точку в тот самый момент, когда он попадает на слой люминофора. **Ускоряющая система** обычно объединяется с фокусирующей. Она состоит из двух металлических пластин с отверстиями для прохождения пучка, установленных перпендикулярно его оси. На эти две пластины подается достаточно высокий положительный потенциал, благодаря чему электроны приобретают необходимую скорость; как правило, ускоряющие потенциалы достигают нескольких тысяч вольт.

Рассмотренная конструкция электронной пушки обладает тем преимуществом, что позволяет объединить все ее элементы в виде единого узла, размещаемого внутри корпуса ЭЛТ. Существуют и другие типы пушек, фокусирующие системы которых выполняются в виде катушек, устанавливаемых на внешней поверхности трубки. Подобные конструкции называют **системами с электромагнитной фокусировкой**, отличая их таким образом от более распространенных электростатических систем, описанных в настоящем разделе. Электромагнитный метод фокусировки позволяет добиться более высокого качества изображения по сравнению с электростатическим, однако в графических дисплеях в основном применяется последний, поскольку электростатические системы значительно дешевле в изготовлении.

Отклоняющая система

Горловина трубки охвачена, словно ярмом, набором катушек, которые составляют часть отклоняющей системы, направляющей электронный пучок в заданные точки экрана ЭЛТ. Как

правило, используются две пары катушек, одна из которых отклоняет пучок в горизонтальном направлении, а вторая — в вертикальном. Основное требование, предъявляемое к отклоняющей системе, — это максимально высокая скорость переноса пучка из точки в точку, поскольку ее быстродействие непосредственно определяет объем информации, которая может быть воспроизведена на экране без видимого мерцания. Для того чтобы пучок отклонялся с нужной скоростью, в катушках должны протекать достаточно большие токи. Они вырабатываются усилителями, которые являются важным элементом отклоняющей системы; эти усилители преобразуют малые напряжения, выдаваемые дисплейными контроллерами, в токи необходимой амплитуды.

В свою очередь дисплейный контроллер генерирует напряжения для отклоняющей системы в соответствии с цифровыми сигналами, поступающими от ЭВМ. Обычно эти сигналы задают значения координат, которые переводятся в напряжения с помощью **цифро-аналоговых преобразователей (ЦАП)**. Для того чтобы вычертить на экране вектор, т. е. определенным образом ориентированный отрезок прямой, на горизонтальную и вертикальную пары отклоняющих катушек необходимо подать линейно изменяющиеся напряжения. Их получают с помощью различных методов, два из которых описываются ниже.

1. Интеграторы. Интегратором называют электронную цепь, которая при подаче на ее вход напряжения постоянной величины вырабатывает на выходе напряжение, возрастающее или убывающее с постоянной скоростью. Таким образом, если величины Δx и Δy , определяющие вектор, преобразовать в напряжения и подать на входы двух интеграторов, последние будут генерировать требуемые для отклонения электронного пучка линейно изменяющиеся напряжения.

2. Цифровые схемы. Быстродействующий цифровой генератор векторов может быть построен, например, на основе схемно реализованного ЦДА, объединенного с парой цифро-аналоговых преобразователей. Всякий раз, когда на вход подобного устройства поступают новые значения координат x и y , эти величины переводятся в отклоняющие напряжения, и на экране высвечивается очередная точка.

Люминофоры

Выбор люминофоров, применяемых в трубках графических дисплеев, определяется их спектральными характеристиками и длительностью послесвечения. Последний параметр определяется как время, за которое яркость излучения спадает до одной десятой начального значения. В идеальном случае послесвече-

ние должно составлять около 100 мс или несколько меньше, и изображение, регенерируемое с частотой 30 Гц, при его перемещении смазывается практически незаметно. Что же касается спектрального состава излучения, то наиболее предпочтительным считается белый цвет, особенно для тех приложений, в которых информация выделяется темным на светлом фоне. Люминофор должен отвечать и ряду других требований: обладать малой зернистостью, что обеспечивает необходимую разрешающую способность, высокой светоотдачей, т. е. способностью к преобразованию электрической энергии в световую, и устойчивостью к возгоранию при длительной бомбардировке электронами.

В попытках улучшить те или иные из перечисленных характеристик было опробовано множество различных люминофоров, в состав которых в разных сочетаниях входили кальций, кадмий и цинк, а также малые количества редкоземельных элементов. Все люминофоры имеют стандартную маркировку — например, P1, P4, P7 и т. д. В графических дисплеях наиболее широкое распространение получили люминофор P7, излучающий голубой цвет и обладающий достаточно длительным послесвечением зеленого оттенка, и люминофор P31, который имеет зеленое излучение и значительно более короткое послесвечение. В обычных черно-белых телевизионных трубках чаще всего используется P4, люминофор с белым излучением примерно той же продолжительности, что и у P31. Существуют люминофоры и со значительно более длительным послесвечением, чем у любого из упомянутых, но из-за того, что на них невозможно получить четкие изображения движущихся объектов, применяются они сравнительно редко.

ЭЛТ со сквозным пучком

Обычные ЭЛТ могут давать изображения лишь одного цвета, зависящего от состава люминофора. Однако для векторных графических дисплеев были разработаны и многоцветные трубки. В них используются покрытия из нескольких слоев различных люминофоров, и выбор нужного цвета осуществляется путем подачи соответствующего ускоряющего напряжения (заметьте, что в одноцветных ЭЛТ оно не меняется).

По своей конструкции трубки со сквозным пучком очень похожи на обычные; новым элементом в них является только многослойный люминофор (чаще всего он состоит из двух слоев — первый, зеленого свечения, наносится непосредственно на экран, а поверх него накладывается слой, испускающий красный свет). Если энергия электронного пучка относительно невелика, он способен возбуждать лишь свечение верхнего слоя люминофо-

ра, и в этом случае на экране воспроизводятся только линии красного цвета. Когда ускоряющее напряжение повышается и скорость электронов увеличивается, пучок, не задерживаясь, проходит сквозь первый слой и возбуждает второй, испускающий зеленый свет. В результате отдельные элементы изображения приобретают зеленый оттенок. Описанным способом получают изображения, содержащие ограниченный набор цветов — обычно это красный, оранжевый, желтый и зеленый.

При создании ЭЛТ со сквозным пучком наиболее сложной проблемой является реализация механизма изменения цвета, с помощью которого амплитуда ускоряющего напряжения модулируется в достаточно широком диапазоне. Следует учитывать, что вариации ускоряющего напряжения сказываются при работе отклоняющей системы, ввиду чего их необходимо компенсировать. Должны быть предусмотрены специальные аппаратные или программные средства, которые позволяли бы выдерживать определенный интервал между сменами цветов, необходимый для установления напряжений. Для того чтобы эти переходы не происходили слишком часто, вызывая мерцание изображения на экране, приходится поочередно выводить сначала все его красные элементы, затем, изменив ускоряющее напряжение, все желтые элементы, и так по всем имеющимся в ЭЛТ цветам.

ЭЛТ с теневой маской

ЭЛТ с теневой маской, используемые в большинстве цветных телевизионных приемников и мониторов, обеспечивают воспроизведение значительно более широкого спектра цветов, нежели ЭЛТ со сквозным пучком. Устройство трубки такого типа показано на рис. 20.24. В непосредственной близости от покрытой люминофором внутренней поверхности экрана ЭЛТ смонтирована металлическая пластина, именуемая **теневой маской**. В этой пластине проделано множество отверстий, причем их горизонтальные ряды сдвинуты относительно друг друга на половину шага, так что каждая тройка отверстий образует правильный треугольник. В трубке с теневой маской установлена не одна, как в обычной ЭЛТ, а три отдельные электронные пушки, торцы которых также составляют правильный треугольник, спереди напоминающий греческую букву «дельта». Электронный пучок каждой из пушек возбуждает свечение только одной из трех цветовых составляющих изображения — красной, зеленой или синей.

Отклоняющая система ЭЛТ воздействует одновременно на все три электронных пучка, сводя их в одну точку — фокус, расположенный в плоскости маски. Если фокус приходится на

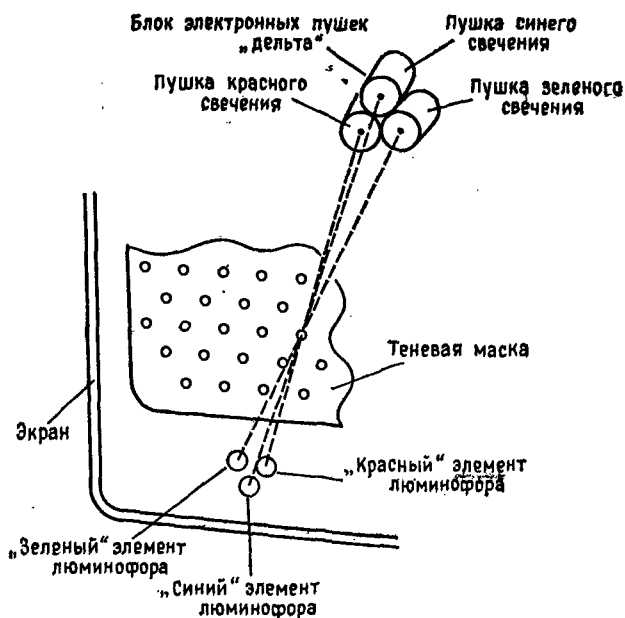


Рис. 20.24. ЭЛТ с теневой маской.

отверстие маски, пучки проходят сквозь него и попадают на поверхность люминофора. Поскольку, однако, они испускаются тремя разными источниками, то те точки люминофора, куда попадают лучи, оказываются на некотором расстоянии друг от друга. Этот эффект и используется — люминофор, с большой точностью накладываемый на внутреннюю поверхность экрана, представляет собой мозаику, содержащую элементы трех типов, которые испускают лучи красного, зеленого и синего цветов. Тройки таких элементов располагаются напротив всех отверстий маски, причем так, что в каждый из них может попасть пучок только из одной пушки. Таким образом, роль маски состоит в том, чтобы загородить элементы люминофора с определенным цветом свечения от всех электронных пучков, кроме того, который предназначен для их возбуждения. Благодаря этому в любой точке экрана можно регулировать долю каждой из трех цветовых составляющих, варьируя токи электронных пучков, испускаемых соответствующими пушками.

С тех пор как в 1950 г. фирма RCA впервые продемонстрировала действующую телевизионную трубку с теневой маской, технология изготовления подобных ЭЛТ постоянно совершенствовалась. Тем не менее и теперь они существенно дороже одноцветных трубок и уступают им по всем показателям, обладая

лишь одним преимуществом — способностью воспроизводить цветные изображения. ЭЛТ с теневой маской имеют более низкую разрешающую способность и меньшую светотдачу. И то и другое объясняется введением маски: разрешение ограничивается расстоянием между просверленными в ней отверстиями, а светотдача мала из-за того, что маска задерживает значительную часть электронов. Впрочем, ценой существенного увеличения ускоряющего напряжения удастся получить изображения, приближающиеся по яркости к изображениям на экранах одноцветных ЭЛТ.

Еще одной проблемой, характерной лишь для трубок с теневой маской, является обеспечение **схождения пучков**. Исключительно сложно настроить все три пушки и отклоняющую систему так, чтобы электронные пучки перемещались совершенно синхронно и постоянно пересекали плоскость маски в одной общей точке. Между тем, если пучки, проходя сквозь отверстие маски, несколько разойдутся, полученное на экране изображение будет напоминать плохой типографский оттиск со смещенными клише разных оттенков. Очень часто требуемой точности схождения удастся достигнуть лишь в ограниченной зоне экрана.

Перечисленные факторы — сложность получения высокой точности схождения, сравнительно низкая разрешающая способность и малая светотдача — стали причиной того, что работчики графических векторных дисплеев редко применяют в них ЭЛТ с теневой маской.

20.18. ГРАФИЧЕСКИЕ УСТРОЙСТВА С ЗАПОМИНАНИЕМ ИЗОБРАЖЕНИЯ

Основными недостатками векторных дисплеев на базе ЭЛТ с регенерацией изображения являются их высокая стоимость и заметное мерцание экрана, возникающее при выводе сложных многоэлементных композиций. Стремлением преодолеть эти проблемы объясняется создание графических устройств, обладающих встроенным механизмом памяти. Среди приборов этого типа наибольшую популярность приобрели **запоминающие электронно-лучевые трубки (ЗЭЛТ)** с прямым копированием изображения; начинают применяться также плазменные панели и дисплеи со сканирующим лазером.

Запоминающие ЭЛТ с прямым копированием изображения

Внешне ЗЭЛТ с прямым копированием действуют, подобно обычным ЭЛТ, с чрезвычайно длительным послесвечением люминофора. Линию, прочерченную на экране такой трубки, мож-

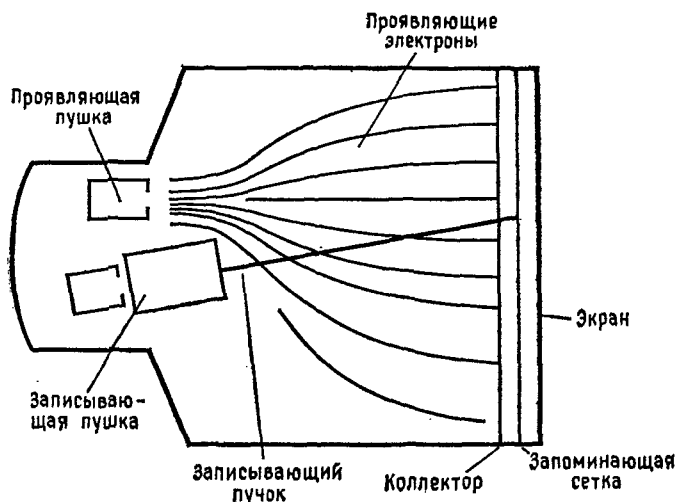


Рис. 20.25. Запоминающая трубка с прямым копированием изображения.

но отчетливо наблюдать в течение по меньшей мере часа, пока она не станет окончательно неразличимой. По внутреннему устройству ЗЭЛТ также напоминает обычную ЭЛТ, поскольку в ней используются такая же электронная пушка и экран, покрытый изнутри люминофором. Однако в запоминающей трубке электронный пучок направляется не на сам люминофор, а на установленную позади него проволочную сетку с очень мелкими ячейками. На сетку наводится положительный заряд определенной конфигурации, которая копируется на люминофоре потоком электронов, непрерывно испускаемых вторым источником, называемым **проявляющей пушкой**. Основные детали конструкции ЗЭЛТ изображены на рис. 20.25.

Позади запоминающей сетки стоит вторая, именуемая **коллектором**; основная ее функция — придание большей равномерности потоку проявляющих электронов. Эти электроны, обладающие малой скоростью, проходят через коллектор и устремляются к положительно заряженным участкам запоминающей сетки, но отталкиваются другими ее участками. Часть электронов все же проникает сквозь запоминающую сетку и попадает на поверхность люминофора. Для того чтобы увеличить энергию этих сравнительно медленно движущихся электронов и тем самым повысить яркость изображения, на экране трубки создается высокий положительный потенциал за счет напряжения, подводимого к тонкому слою алюминия, напыленному на **внутреннюю поверхность экрана под люминофором**,

Однако до того момента, как проявляющие электроны пройдут сквозь запоминающую сетку, они перемещаются с очень низкой скоростью и в силу этого слабо взаимодействуют с распределенным по сетке зарядом. Данное обстоятельство обуславливает основной недостаток ЗЭЛТ: сложность нейтрализации заряда, что необходимо для стирания изображения. Чаще всего стирание осуществляется путем подачи на запоминающую сетку в течение одной-двух секунд положительного напряжения. В результате заряд удаляется, но возникает и побочный эффект в виде неожиданного для наблюдателя сияния, охватывающего всю поверхность экрана. Пожалуй, сложность стирания следует считать главным недостатком ЗЭЛТ, поскольку это препятствует применению подобных приборов для оперативного воспроизведения графической информации. Есть у ЗЭЛТ и другие недостатки, в частности невозможность получения высокой контрастности из-за того, что проявляющие электроны ускоряются под действием сравнительно небольшого напряжения, а также постепенное ухудшение четкости изображения вследствие постоянно возрастающей яркости фона. Возникновение фона обусловлено попаданием на запоминающую сетку малых порций заряда, который наводится проявляющими электронами, отраженными от экрана.

ЗЭЛТ уступают обычным ЭЛТ с регенерацией изображения и по ряду других показателей. В них используются только люминофоры зеленого свечения; яркость всех линий, выводимых на экран, одинакова. До недавних пор ЗЭЛТ имели сравнительно небольшие экраны. В настоящее время начат выпуск запоминающих трубок с экранами в 19 и даже 25 дюймов по диагонали. Малые ЗЭЛТ удобны тем, что экраны у них изготавливаются плоскими, в то время как большие трубки имеют выпуклые экраны. В некоторых дисплеях с ЗЭЛТ предусмотрена возможность регенерации ограниченного числа векторов.

Плазменная панель

Устройство для воспроизведения изображений, получившее название плазменной панели, имеет необычную для дисплеев конструкцию. Картина на поверхности такого устройства высвечивается точка за точкой, причем, раз загоревшись, эти точки продолжают светиться и дальше. В этом отношении плазменная панель напоминает ЗЭЛТ, хотя устройство их совершенно различно.

Основные элементы плазменной панели показаны на рис. 20.26. Она состоит из двух стеклянных пластин, на внутреннюю поверхность которых нанесены тонкие, расположенные на небольшом расстоянии друг от друга полоски золота, играю-

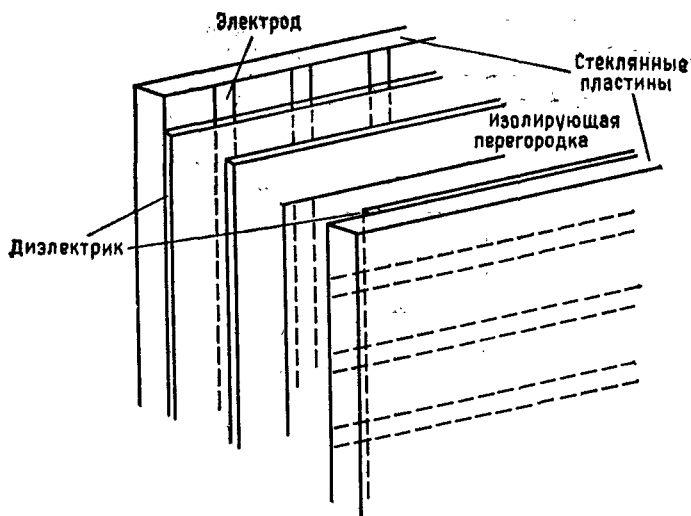


Рис. 20.26. Плазменная панель.

щие роль **электродов**. Эти полосы покрыты сверху слоем диэлектрика. Герметически закрытый зазор между электродами, ширина которого не превышает нескольких десятков микрометров, заполнен газовой смесью на основе неона. При подаче на электроды определенным образом подобранных потенциалов газ, заключенный в панели, начинает вести себя так, будто он разделен на микроскопические ячейки, действующие независимо друг от друга. С помощью весьма тонкого механизма формирования напряжений отдельные выбранные ячейки можно заставить светиться, за счет чего на экране возникает требуемое изображение. Ячейка возбуждается, когда на электроды подается «зажигающий» импульс напряжения. В газовой смеси внутри ячейки начинается тлеющий разряд, который очень быстро развивается и вызывает достаточно яркое свечение. Это свечение не пропадает, если к ячейке подводится напряжение, колеблющееся с высокой частотой; форма **поддерживающего напряжения** показана на рис. 20.27. Очень важно, что при соответствующем подборе амплитуды поддерживающего сигнала не зажженные ранее ячейки не возбуждаются. Таким образом, каждая ячейка функционирует как **бистабильный**, т. е. имеющий два устойчивых состояния, элемент.

Ячейки загораются, когда на поддерживающий сигнал накладывается короткий импульс напряжения; это можно делать выборочно, изменяя сигналы только на той паре электродов, на пересечении которых находится требуемая ячейка. Точно таким

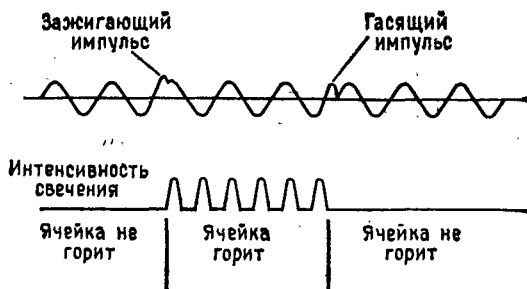


Рис. 20.27. Поддерживающий сигнал, используемый в плазменной панели (вверху), и соответствующие изменения интенсивности свечения ячейки (внизу).

же способом можно гасить зажженные ячейки, снижая амплитуду поддерживающего напряжения на соответствующих электродах. Итак, в плазменной панели обеспечивается как селективное воспроизведение, так и селективное стирание информации; время выполнения этих операций составляет около 20 мкс на ячейку. Быстродействие устройства можно повысить, зажигая и гася по несколько ячеек одновременно.

Плазменная панель дает очень устойчивое изображение без малейшего мерцания. По сравнению с ЭЛТ, имеющей экран того же размера, она более компактна. Однако у нее есть и недостатки. Прежде всего это низкая разрешающая способность (порядка 20 точек на см) и повышенные требования к точности работы механизма записи и стирания информации. Конструкция плазменной панели предопределяет наличие внутренней памяти, однако она не может использоваться столь же гибко, как, например, память в виде буфера образа. В настоящее время стоимость цифровых запоминающих устройств упала до такой степени, что даже растровые дисплеи оказываются дешевле плазменных панелей. По этой причине плазменные панели не нашли широкого распространения и применяются лишь в отдельных моделях современных дисплеев.

Дисплей со сканирующим лазером

Дисплей со сканирующим лазером относится к числу немногих устройств визуализации, обладающих одновременно и большими экранами, и высокой разрешающей способностью. На таком дисплее могут воспроизводиться изображения, достигающие размеров 3 на 4 фута, и при этом точки раstra остаются сравнительно небольшими — их диаметр не превышает 1/100 дюйма. Подобные устройства могут использоваться для

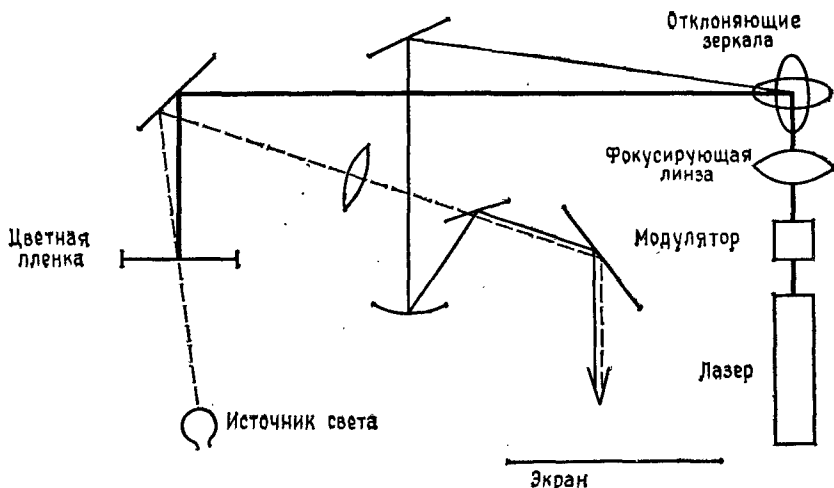


Рис. 20.28. Траектории лучей в дисплее со сканирующим лазером.

————— луч записи изображения; ————— луч курсора; ---- луч воспроизведения изображения.

демонстрации карт, воспроизведения высококачественных текстов и сложных электронных схем.

Принцип работы устройства весьма прост: луч лазера отклоняется парой зеркал, в результате чего он описывает заданную траекторию, двигаясь по кадру цветной пленки. Сам материал пленки делается обычно прозрачным, а луч лазера оставляет на нем темный след. С помощью специальной оптической системы полученное таким способом изображение проецируется на большой экран. Для вывода другого изображения достаточно продвинуть пленку вперед, чтобы под лучем лазера оказался новый кадр.

Отклоняющие зеркала очень миниатюрны и легки; углы их поворота задаются электрическими сигналами, поступающими от дисплейного контроллера. Колебания зеркал устраняются за счет введения сложных корректирующих фильтров. Система может работать и в интерактивном режиме, для чего она оснащается вспомогательным лазером, луч которого, не оставляя следа на пленке, проецируется на экран в качестве курсора. Основные элементы конструкции описанного устройства изображены на рис. 20.28.

Кроме плазменной панели и дисплея со сканирующим лазером, за последние годы было предложено много других устройств, предназначенных для получения высококачественных изображений, генерируемых с помощью ЭВМ. Однако ни одно

из них так и не было по-настоящему внедрено в практику. До сих пор разработчики отдают предпочтение ЭЛТ и ЗЭЛТ, столь хорошо зарекомендовавшим себя в машинной графике. Поэтому оставшаяся часть главы посвящена обсуждению вопросов аппаратной реализации дисплеев, в которых применяются именно эти приборы.

20.19. ДИСПЛЕЙ С ЗАПОМИНАЮЩЕЙ ТРУБКОЙ

На рис. 20.29 приведена фотография типичного дисплея с запоминающей трубкой марки Tektronix 4006-1. Он оснащен ЗЭЛТ с экраном размером 7 на 10 дюймов и встроенной алфавитно-цифровой клавиатурой. Выводимый на экран растр имеет 1024 позиции по горизонтали и 760 позиций по вертикали.

Сигналы, управляющие работой ЗЭЛТ, генерируются дисплейным контроллером в соответствии с данными, выдаваемыми ЭВМ. Эти данные представляют собой последовательность команд, каждая из которых задает положение одного элемента изображения. Например, для того чтобы высветить на экране отдельную точку, в контроллер нужно заслать ее координаты по осям x и y . Контроллер преобразует эти числа в соответствующие напряжения, которые подаются на отклоняющую систему трубки, в результате чего электронный пучок фиксируется в требуемой позиции. Далее энергия электронов на очень короткое время повышается, чтобы образ точки запечатлелся на запоминающей сетке. Аналогичным способом можно воспроизводить и сложные изображения. Для этого они разбиваются на отдельные точки, которые последовательно выводятся на экран дисплея.

С целью сокращения объема вычислений и увеличения пропускной способности в большинстве дисплеев с запоминающими трубками предусматривается вывод не только точек, но и векторов, т. е. отрезков прямых линий. Для этого от ЭВМ должны поступать координаты обоих концов вектора; дисплейный контроллер устанавливает луч в позицию, соответствующую одной из крайних точек, и затем перемещает его по прямой во вторую крайнюю точку. Направление движения луча задается генератором векторов, который формирует линейно изменяющиеся напряжения, подаваемые на отклоняющую систему.

На практике в командах построения векторов оба конца отрезка не задаются. Вместо этого вектор проводится из текущей позиции луча, т. е. того положения, которое луч занял после вывода на экран предыдущей точки или вектора. Использование такого метода упрощает воспроизведение ломаных линий; для того чтобы вывести вектор, не соединенный одним своим концом с предыдущим, достаточно перед командой вычер-

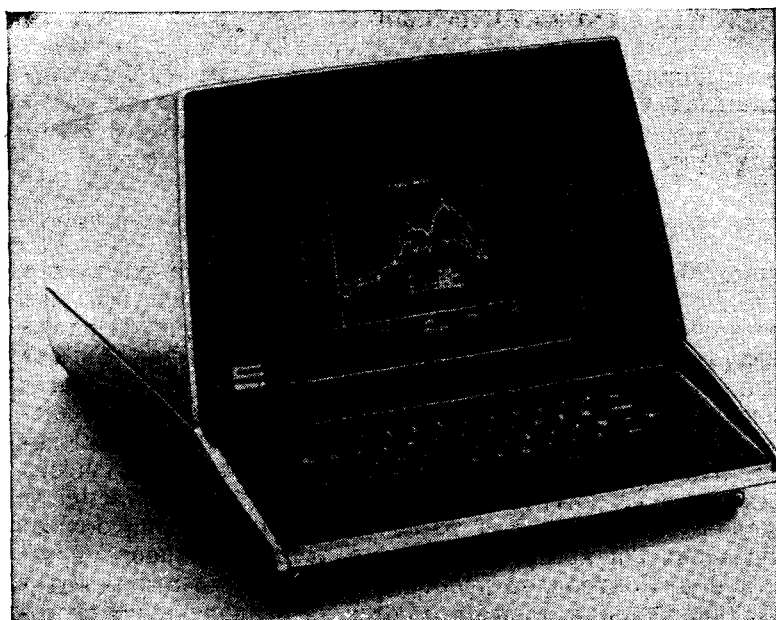


Рис. 20.29. Дисплей 4006-1 фирмы Tektronics (© 1983 Tektronics, Inc., фотография воспроизведена с разрешения фирмы).

чивания вектора поместить команду, которая переводит луч из текущей позиции в одну из крайних точек нового отрезка.

Для дисплея с запоминающей трубкой, в которой значения координат представлены в виде 10-разрядных положительных целых двоичных чисел, в каждой команде под данные должно быть отведено 20 разрядов. По меньшей мере еще один разряд необходим для записи кода операции, определяющего тип команды. Чтобы уменьшить объем информации, пересылаемой за один прием в дисплейный контроллер, команды делят на 7-разрядные байты. Это дает возможность передавать команды для дисплея в том же последовательном асинхронном режиме, который применяется при передаче алфавитно-цифровой информации для текстовых терминалов. Почти все дисплеи с запоминающими ЭЛТ рассчитаны на прием закодированных команд в указанном режиме, что значительно облегчает проблему их сопряжения с ЭВМ.

При использовании 7-разрядных управляющих байтов система команд дисплея несколько усложняется. В качестве примера на рис. 20.30 приведен набор команд дисплея Tektronix 4006-1. Благодаря введению двух команд, ENTER GRAPHICS

0011101		ENTER GRAPHICS MODE						
0011111		LEAVE GRAPHICS MODE						
01	у(старшие разряды)	11	у(младшие разряды)	01	х(старшие разряды)	10	х(младшие разряды)	VECTOR

Рис. 20.30. Базовая система команд дисплея с запоминающей трубкой фирмы Tektronix.

Команда VECTOR, следующая сразу за командой ENTER GRAPHICS MODE, не определяет вычерчиваемый вектор, а лишь задает текущую позицию луча.

MODE (перейти в графический режим) и LEAVE GRAPHICS MODE (выйти из графического режима), этот дисплей может действовать и как алфавитно-цифровой, и как графический терминал. Когда дисплей функционирует в графическом режиме, управляющие байты интерпретируются как составные части команды построения векторов. Если поступает команда LEAVE GRAPHICS MODE, режим меняется и управляющие байты начинают интерпретироваться как коды символов ASCII (Американский стандартный код обмена информацией). При этом каждый очередной символ высвечивается в позиции, соответствующей текущему положению луча, после чего тот перемещается вправо на ширину выведенного символа. Для того чтобы можно было вновь вернуться к построению точек и векторов, по завершении вывода текстовой информации необходимо подать команду ENTER GRAPHICS MODE.

В дисплее Tektronix 4006-1 команды построения векторов имеют формат, позволяющий получать очень компактные описания изображений, содержащие в основном короткие векторы.

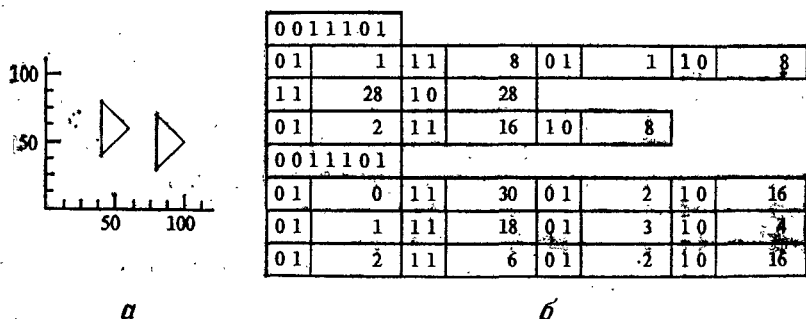


Рис. 20.31. Простое изображение (а) и его закодированное представление (б) для дисплея с запоминающей трубкой; использованы команды, приведенные на рис. 20.30.

Если координаты двух соседних точек отличаются только содержимым пяти младших разрядов, старшие разряды в контроллер не пересылаются. Последовательность команд, кодирующих простое изображение, высвечиваемое на экране запоминающей трубки, представлена на рис. 20.31.

20.20. ВЕКТОРНЫЕ ДИСПЛЕИ С РЕГЕНЕРАЦИЕЙ ИЗОБРАЖЕНИЯ

До 1968 г., когда начали выпускаться первые модели дисплеев с запоминающими трубками, практически во всех графических терминалах устанавливались ЭЛТ с регенерацией изображения. И в настоящее время подобные терминалы, несмотря на то что они сложнее по конструкции и дороже дисплеев с запоминающими трубками, остаются наиболее распространенным типом устройств, используемых в интерактивной машинной графике. Такая их популярность объясняется в первую очередь тем, что они позволяют отслеживать динамику изменения изображений. Это совершенно необходимо при решении многих прикладных задач, например при воспроизведении результатов моделирования или при работе с интерактивными графическими программами.

Преобразование сигналов, генерируемых ЭВМ, в напряжения, подаваемые в отклоняющую систему ЭЛТ, для векторных дисплеев с регенерацией изображения, как и для дисплеев с запоминающими трубками, осуществляется в дисплейных контроллерах. Во многих отношениях последние очень схожи с контроллерами дисплеев с запоминающими трубками; они точно так же интерпретируют команды построения векторов, используя текущую позицию пучка в качестве начальной точки отрезка.

Какие же отличительные особенности присущи контроллерам дисплеев с регенерацией изображения? Первая и главная из них — это значительно более высокое быстродействие. Картина на экране ЭЛТ будет воспроизводиться без мерцания только при том условии, что ее закодированное описание будет подаваться на трубку не менее 30 раз в секунду. Предположим, что изображение составлено из 5000 векторов, и все они должны быть построены дисплейным контроллером в течение одного цикла регенерации продолжительностью $1/30$ с. Это означает, что контроллеру необходимо иметь быстродействие, обеспечивающее обработку $150\,000$ (30×5000) команд построения векторов в секунду. Разумеется, такая цифра лежит далеко за пределами возможностей обычных линий связи, действующих в режиме асинхронной последовательной передачи данных.



2	I		Координата x	Смещение по горизонтали
3	I		Координата y	Смещение по вертикали
4	I	D	$\pm \Delta x$	Составляющая по оси x Составляющая по оси y } Вектор
5	I	D	$\pm \Delta y$	

Рис. 20.32. Команды вывода точек и построения векторов для дисплея с регенерацией изображения.

$I = 0$ соответствует нулевой интенсивности, $I = 1$ — нормальной; $D = 1$ указывает на построение вектора. В векторных командах из двух слов первое содержит $D = 0$, второе — $D = 1$. Горизонтальные или вертикальные векторы вычерчиваются посредством одинарной команды с $D = 1$.

Могут быть предложены два способа повышения производительности контроллеров для дисплеев с регенерацией изображения. Во-первых, это использование многоразрядных информационных шин, соединяющих контроллер с памятью ЭВМ; во-вторых, это более эффективная организация доступа к памяти. Как правило, ЭВМ, комплектуемые дисплеями с регенерацией изображения, снабжены средствами вывода информации в 16-разрядном параллельном формате. Их можно было бы использовать с помощью программы, выполняемой центральным процессором (ЦП), которая пересылала бы в дисплей по одной команде в ответ на каждый поступающий от него запрос. Однако более целесообразно сделать так, чтобы дисплейный контроллер, не загружая ЦП, работал в режиме прямого доступа к памяти. При этом он должен считывать из памяти предназначенные для него данные, не пользуясь услугами ЦП, а попросту «перехватывая» у последнего цикл памяти всякий раз, когда дисплей готов принять очередную команду. В контроллере имеется регистр адреса, содержимое которого после выборки каждой команды получает соответствующее приращение; таким образом, в регистре формируется адрес следующей команды. Вся совокупность команд, которую называют дисплейным файлом, занимает непрерывный массив ячеек памяти.

Обычно векторные дисплеи могут выполнять как операции вычерчивания векторов, так и вывода отдельных точек. На рис. 20.32 представлены типичные форматы команд построения векторов и точек. Заметим, что в них задаются относительные значения координат, определяющие смещения пучка от его текущей позиции; предусмотрен также бит, управляющий интенсивностью свечения. Задавая точку с нулевой интенсивностью,

2	0	40
3	0	40
4	00	20
5	11	20
4	00	-20
5	11	20
5	11	-40
2	0	80
3	0	30
4	00	20
5	11	20
4	00	-20
5	11	20
5	11	-40

Рис. 20.33. Дисплейный файл, описывающий изображение на рис. 2.31, а; использованы команды в формате, показанном на рис. 20.32.

можно переместить пучок в новую текущую позицию, а вектор нулевой интенсивности позволяет сдвинуть пучок на требуемое расстояние так, чтобы он не оставил видимого следа на экране. На рис. 20.33 приведена последовательность команд, с помощью которой описывается изображение, показанное на рис. 20.31, а.

Для того чтобы можно было реализовать такое важное преимущество ЭЛТ, как возможность динамического преобразования изображений, нужно, чтобы дисплейный контроллер не только обладал высоким быстродействием, но и был способен гибко менять последовательность выполняемых операций. Уже отмечалось, что динамическое преобразование необходимо либо для визуального наблюдения за текущими результатами, выдаваемыми программой в ходе ее выполнения, либо для обеспечения пользователя интерактивной программы средствами, позволяющими оперативно воздействовать

на ее ход. Все корректировки изображения осуществляются путем изменения содержимого дисплейного файла. Часто бывает весьма непросто быстро внести требуемые изменения в большой массив команд дисплея, занимающий непрерывную область памяти. Однако, если дисплейный файл расчленить на ряд порознь расположенных массивов, задача несколько упрощается. Поэтому в системах команд большинства дисплейных контроллеров имеется команда, позволяющая менять содержимое регистра адреса. Ее называют командой перехода, поскольку по своему действию она аналогична командам перехода или передачи управления, используемым в ЦП.

Дисплейный контроллер, обладающий возможностью самостоятельно устанавливать с помощью команды перехода свой адресный регистр, может, разумеется, заслать в него также начальный адрес дисплейного файла после того, как тот будет исчерпан. В результате команды, поступающие в контроллер, образуют замкнутый цикл, выполняемый вообще безо всякого вмешательства процессора; более того, благодаря этому можно, не прерывая работу дисплея, производить и модификацию дисплейного файла. Контроллеры, способные функционировать совершенно независимо от ЦП, называют обычно дисплейными процессорами. В настоящее время подобными контроллерами осна-

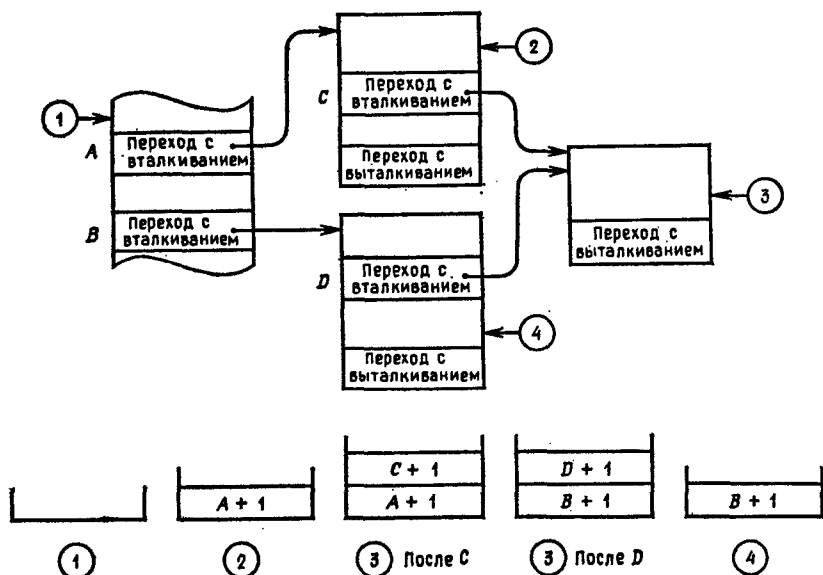


Рис. 20.34. Использование команд перехода с вталкиванием и перехода с выталкиванием для вызова вложенных подпрограмм.

щается большинство векторных дисплеев с регенерацией изображения.

Кроме основной команды перехода вводится, как правило, еще и команда перехода к подпрограмме. С помощью этой команды в регистр адреса дисплейного процессора можно заслать начальный адрес требуемого фрагмента дисплейного файла; команда возврата из подпрограммы восстанавливает прежнее содержимое регистра, позволяя, таким образом, возвратиться к выполнению вызвавшей программы. Для того чтобы подпрограммы могли в свою очередь вызывать другие подпрограммы, адреса возвратов запоминаются в стеке. Последний представляет собой массив ячеек памяти, снабженный указателем, содержащим адрес вершины стека, т. е. той его ячейки, которая была занята позже остальных. Когда выполняется команда перехода к подпрограмме, адрес возврата «вталкивается» в стек, а содержимое его указателя увеличивается на единицу и становится равным адресу той ячейки, куда был только что помещен адрес возврата. Действующую таким образом команду перехода к подпрограмме часто называют **переходом с вталкиванием**. Для выхода из подпрограммы адрес возврата «выталкивается» из стека, т. е. содержимое его вершины пересылается в регистр адреса; одновременно уменьшается на единицу адрес,

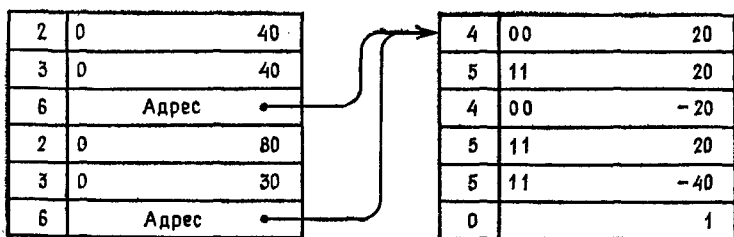


Рис. 20.35. Использование дисплейной подпрограммы для описания изображения, показанного на рис. 20.31, а.

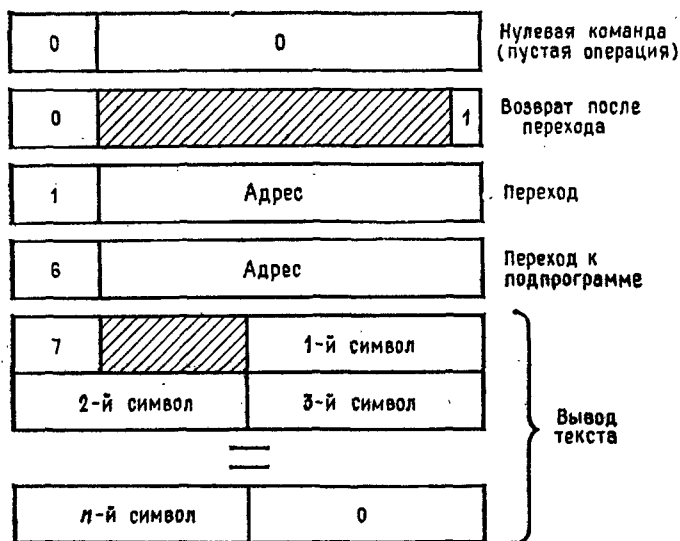


Рис. 20.36. Форматы команд перехода и вывода текстовой информации.

заклученный в указателе стека. Поскольку эта процедура действительно напоминает «выталкивание» содержимого верхней ячейки стека, команду возврата именуют также **переходом с выталкиванием**. Состояние стека на разных этапах выполнения последовательности вложенных подпрограмм иллюстрируется рис. 20.34.

Подобно обычным, дисплейные подпрограммы дают возможность исключить многократную запись повторяющихся цепочек команд; в тех случаях, когда на экран приходится выводить однотипные фигуры, за счет применения подпрограмм иногда удается существенно сократить длину дисплейного файла. Перед переходом к подпрограмме вычерчивания фигуры помеща-

ют две команды, задающие новое текущее положение пучка. На рис. 20.35 показано, как можно использовать подпрограмму для построения изображения, приведенного на рис. 20.31, а.

Естественно предположить, что командами перехода к подпрограмме особенно удобно пользоваться при воспроизведении текстов; действительно, имея для каждого символа отдельную подпрограмму, в дисплейном файле строки символов можно записывать просто как цепочки следующих друг за другом переходов на подпрограммы. Однако такой метод приводит к большому расходу памяти, поскольку приходится тратить по полному 16-разрядному слову на каждый высвечиваемый символ. Более целесообразно дополнить систему команд дисплея **командой вывода текста** и использовать вместо набора подпрограмм **схемный генератор символов**. Применение команды вывода текста позволяет кодировать каждый символ одним полусловом (8-разрядным байтом). Дисплейный процессор пересылает эти байты в генератор символов, и тот формирует их в виде либо наборов коротких штрихов, либо точечных матриц, которые он выбирает из постоянной памяти небольшого объема.

Форматы команд перехода и вывода текста приведены на рис. 20.36. В большинстве современных векторных дисплеев с регенерацией изображения предусмотрены и другие команды, в том числе позволяющие регулировать яркость, выводить короткие отрезки прямых или кривых линий и т. д. Форматы этих дополнительных команд подобны форматам команд из основного набора.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И РОБОТОТЕХНИКА

Г. Хелмс

21.1. ВВЕДЕНИЕ

Термин **искусственный интеллект** обычно используется для обозначения способности вычислительной системы выполнять задачи, свойственные интеллекту человека, например задачи логического вывода и обучения. Термин **робот** — это синоним интеллектуальной вычислительной системы, т. е. системы, в состав которой входят устройства, реагирующие на изменение окружающей среды и изменяющие ее. Теоретические и прикладные вопросы построения и применения роботов объединяются в научно-техническое направление, именуемое робототехникой. В последние годы интерес к вопросам искусственного интеллекта и робототехнике значительно возрос.

21.2. КОМПЬЮТЕР И ИНТЕЛЛЕКТ

Вопреки широко распространенному представлению, что компьютеры могут все, в действительности они почти совсем лишены тех качеств, которые связаны с понятием «интеллект». Компьютеры могут быстро и надежно решать задачи, но эти задачи, а также правила их решения должны быть точно определены. В том случае, если, к примеру, заданы исходные данные, делающие невозможным решение задачи на выбранном для этой цели компьютере, самостоятельно внести необходимые изменения он не может.

Проблема создания систем искусственного интеллекта становится более понятной, если выделить в этих системах два компонента. Один включает объективную, поддающуюся оценке и измерению информацию о событиях и объектах. Реализация этого компонента с помощью компьютера не представляет больших затруднений. Второй компонент системы искусственного интеллекта включает субъективную информацию, т. е. ассоциативные связи, модели распознавания образов, правила вывода, механизмы обучения и моделирования интуиции. Как оказалось, реализация этого компонента в компьютерной системе намного труднее.

Некоторые задачи, содержащие элемент интеллекта, достаточно просты и легко могут быть реализованы с помощью вычислительной техники. Например, пусть даны две строки символов:

„ABCDEF GHIJK“

„ABCDEF GHIJ“

Компьютер сможет быстро определить, что эти две цепочки не одинаковы, выполнив простые действия. Однако ответить на вопрос, в чем заключается это отличие, компьютеру намного труднее. Человек, пользуясь знанием алфавита, понял бы, что в одной из строк пропущена буква. Чтобы то же самое мог сделать компьютер, ему пришлось бы произвести обращение к нескольким подпрограммам, возможно и не связанным с анализом алфавита.

Интеллект человека с помощью интуиции, предположений, накопленного опыта способен заполнить пробелы в имеющихся данных. Компьютер, напротив, должен пользоваться заданной информацией. Человек способен изменить способ решения, если этот способ не дает правильного ответа. Компьютерные системы должны следовать инструкциям, входящим в программы.

Во многих случаях имеют место проявления субъективных свойств интеллекта, которые на первый взгляд кажутся строго объективными. Это можно продемонстрировать на примере задачи распознавания образов. Сопоставление конкретной фотографии человека с его именем (например, Джон Джонс) позволяет компьютеру строить ассоциативную связь с этим человеком, т. е. «узнавать» его. Это случай проявления строго объективной части интеллекта. Если Джон Джонс переоденется или изменит прическу, его еще могут узнать. Однако компьютер не сможет сопоставить новое изображение Джона Джонса со старым.

Рассмотрим другой пример того, как человек может дополнять элементами информацию, которую компьютер интерпретировал бы непосредственно. Строка букв

«Что вы имеете в виду?»

может быть по-разному интерпретирована человеком. Она может быть рассмотрена просто как прямой запрос о дополнительной информации. Ее можно понять как шутливое замечание, как реплику на обвинение или, наконец, как обвинение. Компьютер, даже имея в памяти полный словарь, не смог бы так по-разному интерпретировать эту строку. Действительно, многие задачи искусственного интеллекта включают в себя проблемы интерпретации языка,

Другая проблема искусственного интеллекта основана на разнице в «аппаратуре», используемой человеком и компьютером. Хотя функционирование мозга на клеточном уровне полностью не изучено, ясно, что он действует посредством одно-временной активизации миллионов клеточных «логических устройств». Очень возможно, что человеческий мозг имеет огромное преимущество перед аппаратурой компьютера при реализации характерных свойств интеллекта. Решение проблемы строения человеческого мозга позволило бы начать разработку систем искусственного интеллекта, приближающихся по своим возможностям к человеческому.

21.3. ЭКСПЕРТНЫЕ СИСТЕМЫ

Экспертные системы представляют собой попытку использования в компьютерных системах некоторых механизмов обоснования и принятия решений, свойственных человеку. Экспертные системы строятся из следующих трех элементов: базы данных, базы правил и интерпретатора правил.

Базой данных называется часть памяти, предназначенная для запоминания объективной информации. **Базу правил** иногда называют базой знаний. Этот файл содержит решающие правила, позволяющие оперировать с определенным классом задач. Правила строятся с помощью группы людей, называемых экспертами. Часть системы, называемая **интерпретатором правил**, позволяет манипулировать документальными данными из базы данных с помощью правил, находящихся в базе правил.

Правила, хранящиеся в базе правил, имеют следующую структуру:

IF условия

THEN следствие

Поскольку следствие может быть неоднозначным, то обеспечивается возможность его корректировки. Во время функционирования программ экспертной системы правила из базы правил загружаются в интерпретатор правил, который начинает просматривать информацию из базы данных. Если выбранная единица или набор данных соответствуют условиям правила, результат записывается в базу данных. Затем интерпретатор правил продолжает поиск в базе данных.

Очевидно, что каждое из правил представляет собой подпрограмму, которая, однако, не вызывается посредством указания ее имени или адреса точки входа, а выполняется при появлении определенных типов данных. О подпрограмме такого типа говорят, что она **управляется по образцу**.

Другой подход к построению экспертных систем заключается в определении целей, которые должны быть достигнуты в результате работы системы. Интерпретатор исследует те правила, которые соответствуют или приводят к желаемым целям. Он осуществляет выборку только тех данных, использование которых в условиях правил могло бы привести к желаемому результату. Если ни одно из данных нельзя использовать в условиях и не существует правил, обеспечивающих достижение сформулированной цели, интерпретатор сообщит человеку-оператору о необходимости ввода дополнительных данных или правил.

Экспертные системы с успехом используются при решении задач диагностики в геологии и медицине.

21.4. РОБОТОТЕХНИКА

Робототехника — научно-техническое направление, изучающее вопросы проектирования и применения роботов. В сущности робототехника представляет собой синтез электроники и механики.

В робототехнике кроме задач, встречающихся при построении систем искусственного интеллекта, существует много специфических для этой области проблем. Большая часть этих проблем связана с взаимодействием робота с окружающей средой. Во многих случаях бывает недостаточно только способности робота манипулировать объектами внешнего мира. Робот должен иметь способность получать и интерпретировать информацию об окружающей среде, а также согласовывать свои действия в соответствии с этой информацией. Приведем такой пример. Для того чтобы написать что-либо, мало уметь манипулировать ручкой. Необходимо также уметь контролировать степень давления пера на бумагу, проверять правильность формы написанных букв, а также следить за тем, чтобы перо соприкасалось с листом.

Как можно заключить из изложенного выше, для реализации универсального робота из научной фантастики потребуется еще много лет труда. В настоящее время роботы или создаются только для определенных целей, или могут настраиваться путем перепрограммирования на решение желаемых задач.

21.5. ЗАДАЧИ В РОБОТОТЕХНИКЕ

Задача в робототехнике может быть задана в терминах отдельных движений, необходимых для выполнения желаемых операций, таких, как поднятие объекта с поверхности или затягивание болта. Задача также может трактоваться как цикл, состоящий из отдельных операций.

Такое определение задачи вполне подходит для множества операций, встречающихся в настоящее время при использовании роботов, например при точечной сварке или при снятии деталей со сборочной линии. Такого сорта задачи не отличаются разнообразием вариантов решения, а состоят в повторении однотипных движений. В робототехнических задачах легко выделить параметры, характеризующие выполняемую операцию. Это могут быть, например, величина перемещения руки робота, сила давления при захвате детали и требуемое время сварки.

Если цикл движений робота определен не полностью, решение становится намного более сложным. Например, возможна ситуация, когда деталь на сборочной линии установлена не на своем месте, и сварка не может быть произведена правильно. В другом случае детали могут иметь различную прочность и для их захвата требуется различная сила. Эти примеры иллюстрируют, насколько необходима для робота способность обнаруживать изменения в окружающей обстановке и реагировать на них. Например, в первом примере робот должен иметь возможность перед сваркой установить объект на нужное место. Эту способность робота к реакции на окружающую среду в рамках современного уровня технологии можно обеспечить, используя заранее определенные и записанные в память робота диапазоны допустимых изменений параметров, характеризующих ситуацию. Когда наступает момент выполнения задачи, робот может оценить параметры ситуации. Если эти параметры находятся вне пределов допустимого интервала, задача не выполняется. Робот в этом случае может выполнить ряд вспомогательных действий, как, например, вызвать человека-оператора, убрать деталь со сборочной линии или произвести такие операции над объектом, которые обеспечат допустимые параметры ситуации.

Создание робота, способного выполнять нечетко поставленные задачи, не является в настоящее время главной задачей. Однако разработка программного обеспечения даже для сравнительно простых задач требует больших усилий. В действительности при внедрении роботов для многих производственных ситуаций создание программного обеспечения представляет собой наиболее трудную проблему.

21.6. РОБОТЫ И ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ

В настоящее время робототехника оказалась в центре внимания специалистов и энтузиастов в области персональных компьютеров. Это произошло благодаря тому, что примерно с 1982 г. роботы стали элементом периферийного оборудования для небольших персональных компьютеров,

Типичным примером может служить устройство Microbot Alpha, которое выпускается фирмой Microbot, Inc. Оно представляет собой роботизированную руку, которая выполняет движения относительно пяти осей. Управление механическим устройством захвата, способным поднять груз весом более 0,5 кг, осуществляется с помощью специального кабеля. Благодаря тому что устройство снабжено стандартным последовательным интерфейсом RS-232C, оно может управлять командами, передаваемыми от микрокомпьютера. Кроме того, в его аппаратуру входит контроллер, имеющий блоки как постоянной, так и оперативной памяти, а также микропроцессор, управляющий работой электромоторов робота. Как и во многих других роботах, работа электромоторов осуществляется в дискретном режиме.

Основной шаг в направлении объединения робототехнических устройств с персональными компьютерами был сделан фирмой IBM, выпустившей в 1982 г. устройство 7532 Manufacturing System. Это устройство представляет собой роботизированную руку и использует персональный компьютер фирмы IBM для управления движением механической руки и для создания соответствующего программного обеспечения. Специалисты считают, что в ближайшее время степень интеграции персональных компьютеров и роботов значительно увеличится, поскольку повышение гибкости робототехнических систем за счет использования микропроцессоров позволяет решить некоторые основные проблемы развития промышленных роботов.

СИМВОЛЬНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА¹⁾

Л. Хоэнштейн

22.1. ВВЕДЕНИЕ

Для формирования записей выходной информации ЭВМ или, как говорят, для получения **твердой копии** используют две группы устройств: символьные печатающие устройства и графопостроители. Поясним их различие. Хотя традиционно символьные печатающие устройства печатали только буквы и цифры, а графопостроители рисовали ломаные и кривые, из которых складывался чертеж, рисунок или график,— сейчас обе группы по своим функциям часто перекрывают друг друга. Большинство символьных печатающих устройств позволяет создавать некое подобие графика, используя символы типа *, X и тому подобные. Возможно вырисовывание и некоторого подобия картин.

В свою очередь большинство графопостроителей допускают вывод букв и цифр, поскольку последние представляют простейшие графические формы. В настоящее время создаются устройства, называемые фирмами-изготовителями как **печатающие графопостроители**, поскольку они достаточно эффективно реализуют как функции печати символов, так и отрисовки графических изображений. Вполне вероятно, что по мере совершенствования технологии появятся устройства, которые и печатать текст и рисовать будут одинаково хорошо.

В дальнейшем будем придерживаться следующей меры различия этих типов устройств. Будем считать, что печатающие устройства предназначены для формирования удобочитаемых текстов из букв, цифр и других символов и выполняют эти функции заведомо эффективнее, чем отрисовку линий. В свою очередь графопостроители — это устройства, которые предназначены для отрисовки графических изображений, причем делают они это эффективнее, чем распечатку символьных строк.

¹⁾ Adapted from Computer Peripherals for Minicomputers, Microprocessors, and Personal Computers, by Louis Hohenstein. Copyright © 1980. Used by permission of McGraw-Hill, Inc. All rights reserved.

Отличие печатающих устройств от терминалов и удаленных печатающих устройств

Печатающее терминальное устройство состоит из собственно печатающего устройства и периферийного оборудования. Вся конфигурация в целом служит, как правило, для обеспечения доступа к некоторой удаленной ЭВМ. Например, удаленное печатающее устройство — это печатающее терминальное устройство с интерфейсом для связи с ЭВМ через телефонную линию.

22.2. КЛАССИФИКАЦИЯ ПЕЧАТАЮЩИХ УСТРОЙСТВ

Печатающие устройства, используемые в составе мини- или микроЭВМ, можно объединить в три группы.

Ударные и безударные устройства. В ударных печатающих устройствах символы печатаются традиционным способом, т. е. с помощью печатающей головки, которая бьет по бумаге через красящую ленту. В безударных устройствах символы печатают-



Рис 22.1. Классификация печатающих устройств.

Таблица 22.1. Характеристики печатающих устройств

Тип устройства	Способ печати	Технология печати	Скорость печати ¹⁾	Примечание
Ударные	Посимвольный	Цилиндрический носитель	10 симв./с	Модель 33 фирмы Teletype Пишущая машинка Selectric фирмы IBM
		Сферическая шрифтовая головка	15 симв./с	
	Построчный	Шрифтоноситель типа ромашки или втулки	30—55 симв./с	Высокоскоростная последовательная печать символов с цельным очертанием Высокоскоростная символьная ударная печать
		Матричная головка с ударными проволочками	30—330 симв./с	
Безударные	Построчный	Шрифтовая лента	до 3000 стр./мин	Изменяемый набор символов Изменяемый набор символов
		Шрифтовая цепь	до 2000 стр./мин	
	Посимвольный	Барaban	300—2000 стр./мин	При больших скоростях в строке образуются небольшие смещения букв по вертикали
		Гребенка точно-матричной печати	до 600 стр./с	
Безударные	Посимвольный	Термическая точно-матричная печать	30—120 симв./с	Дешевое бесшумное устройство; требует специальной бумаги; не позволяет получать копии
		Электростатическая точно-матричная печать	160—2000 симв./с	
	Построчный	Струйная печать	30—1000 симв./с	Требуется специальная бумага; печать низкого качества Не требует специальной бумаги; не позволяет получать копии
		Электростатическая матричная печать	300—18 000 стр./мин	
	Построчный	Лазерная или ксерографическая печать	4000—21 000 стр./мин	Требуется высокоскоростное дорогое устройство, требующее квалифицированного обслуживания; позволяет получать весьма объемные распечатки
		Фотооптическая печать	150—1000 стр./мин	Используется как фотонаборная установка со встроенной мннн-ЭВМ

¹⁾ Здесь симв./с — символов в секунду, стр./мин — строк в минуту, стр./с — строк в секунду.

ся без привлечения ударного механизма, например путем местного разогрева специальной чувствительной бумаги или же путем разбрызгивания на бумагу чернил.

Целостные и точечно-матричные символы. Целостные символы похожи на те, которыми печатает обыкновенная пишущая машинка. В этом случае каждый символ отчеканен на обратной стороне печатающего рычага и при печати имеет цельное, связанное очертание без пропусков каких-нибудь элементов. Точечно-матричный символ представляет некоторую комбинацию отдельных точек, которая внешне напоминает этот символ.

Посимвольные и построчные устройства. В посимвольных печатающих устройствах (часто говорят еще о просто **символьных**, или **последовательных, устройствах**) каждый символ печатается практически одновременно, а строка символов выводится последовательно, один символ за другим. В построчных, или **строчных**, устройствах практически одновременно печатается каждая отдельная строка. (В некоторых устройствах, использующих новейшую технологию, например лазерную или ксерографическую, строки также выводятся столь быстро, что практически одновременно печатается целая страница, и тогда говорят о **страничных печатающих устройствах**. В составе мини- и микроЭВМ такие устройства используются редко и в специальных целях, например в фотонаборных системах.)

На рис. 22.1 приведена классификация печатающих устройств с учетом ранее введенных групп, а также с учетом технологии печатания (подробнее об этом будет сказано ниже). В табл. 22.1 сведены характеристики различных типов печатающих устройств.

22.3. НАБОР СИМВОЛОВ

Большинство печатающих устройств, используемых в составе мини- и микроЭВМ, ориентировано на символьные наборы ASCII¹⁾ из 48, 64, 96 или 128 символов.

Наборы из 48 и 64 символов включают широко используемые специальные символы, цифры, пробел, а также заглавные буквы английского алфавита. В число 96-символьного набора ASCII добавлены, кроме того, строчные буквы английского алфавита и некоторые другие специальные символы. В этом наборе не печатаются 2 символа: пробел и забой, а все остальные 94 выводятся на печать.

В полный 128-символьный набор ASCII входят еще 32 особых символа, обычно используемые для организации связи и

¹⁾ ASCII — American National Standard Code for Information Interchange — Американский национальный стандартный код для обмена информацией. — *Прим. ред.*

управления. Как правило, эти символы на печать не выводятся и выполняют некоторые дополнительные функции по управлению терминалом и установлению связи с ним. Однако в некоторых специальных случаях этим 32 кодам могут быть приписаны и какие-нибудь символы для печати; например, буквы не из английского алфавита, для которых нет кода ASCII. Такие дополнительные символы, будь это буквы или специальные значки, используемые при построении графических изображений, могут выводиться на печать наравне с любыми другими 96 символами. Естественно, что когда в качестве дополнительного используется набор специальных символов для отрисовки, то изобразительные возможности печатающего устройства возрастают, а качество выводимых изображений, таблиц, диаграмм и других графических форм существенно улучшается.

22.4. ПОСИМВОЛЬНЫЕ УДАРНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА С ЦЕЛЬНЫМ ОЧЕРТАНИЕМ СИМВОЛОВ

Примером устройств такого типа может служить обычная пишущая машинка. Каждый печатаемый ею символ имеет цельное очертание, поскольку полностью, без каких-нибудь пропусков, вытиснут на определенном рычаге; печать символа происходит разово при нажатии соответствующей клавиши. Как правило, обычные пишущие машинки не используются «в стыке» с ЭВМ, потому что, во-первых, в них отсутствует возможность кодирования клавиш, что необходимо для связи с ЭВМ, и, во-вторых, они достаточно медленны в работе. Тем не менее иногда такие машинки переделывают, встраивая в их клавиатуру соленоиды, а в корпус — специальную схему для возможности управления соленоидами со стороны ЭВМ.

В прошлом классическим для мини- и микроЭВМ было подключение устройства модели 33 фирмы Teletype (рис. 22.2). Эти устройства первоначально создавались для приема и печати сообщений при передаче новостей по телеграфным линиям. То, что они стали использоваться с небольшими ЭВМ, было естественно, поскольку они уже имелись на рынке, были дешевы и просты в подключении. Скорость печати для модели 33 составляет 10 симв./с, что по теперешним временам мало: аналогичные современные устройства, специально разработанные для ЭВМ, имеют скорость 55 симв./с и более.

Собственно, для печати в модели 33 использован вертикальный цилиндр, по поверхности которого в несколько рядов и колонок вытиснены символы (рис. 22.3). Когда на устройство посылается код ASCII, цилиндр поворачивается на нужный угол и при необходимости приподнимается или опускается так, чтобы соответствующий символ оказался напротив бумаги.

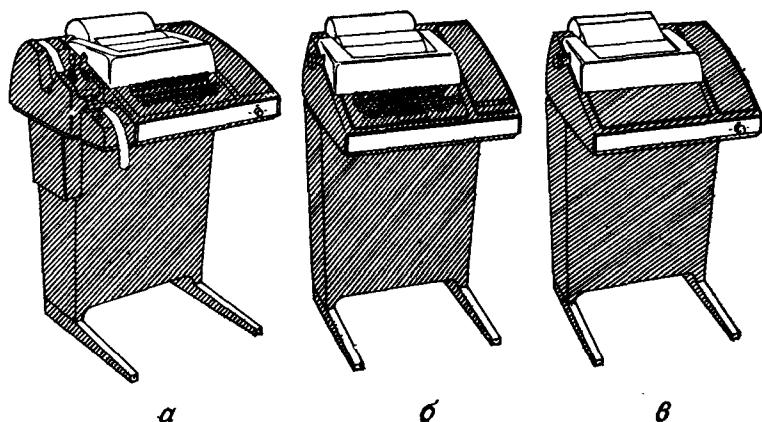


Рис. 22.2. Печатающие устройства фирмы Teletype.

Эти устройства использовались одно время как печатающие устройства для ЭВМ. Многие оригинальные экземпляры модели 33 все еще находят применение. *а* — автоматическое устройство приема-передачи; *б* — устройство только с клавиатурой приема-передачи; *в* — устройство только для чтения (фирма Teletype).

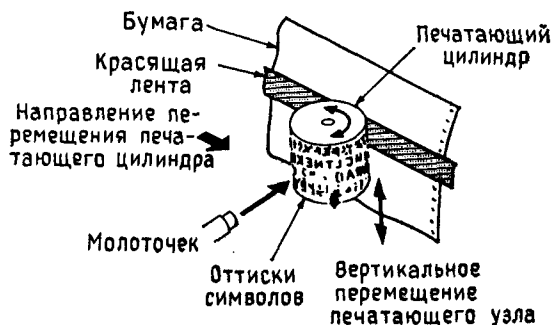


Рис. 22.3. В модели 33 фирмы Teletype использована цилиндрическая печатающая головка.

На поверхности цилиндра оттиснут набор символов. Чтобы обеспечить правильное расположение требуемого символа, цилиндр может поворачиваться вокруг своей оси в приподниматься или опускаться. Вслед за этим по цилиндру ударяет молоточек и производится отпечаток символа.

После этого ударный механизм толкает цилиндр к бумаге, и выставленный символ бьет через красящую ленту. На бумаге в результате остается отпечаток — изображение символа.

Модель 33 фирмы Teletype подключается к небольшой ЭВМ по токовой петле 20 или 60 мА, позволяющей последовательно передавать ASCII-коды. Портом ввода-вывода на 20 или 60 мА обладают многие домашние и небольшие деловые ЭВМ; помимо этого, широко распространены адаптеры, позволяющие подключать модель 33 к порту RS232C. Несмотря на то что

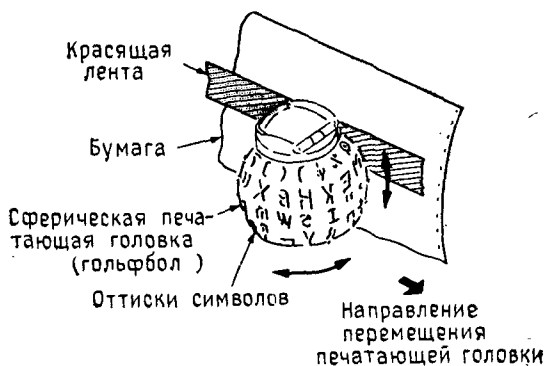


Рис. 22.4. Пишущая машинка Selectric фирмы IBM.

В машинке использована сферическая головка, похожая внешне на мячик для гольфа и являющаяся усовершенствованной версией печатающего цилиндра. Головка поворачивается нужным символом к бумаге и ударяет по ней целиком.

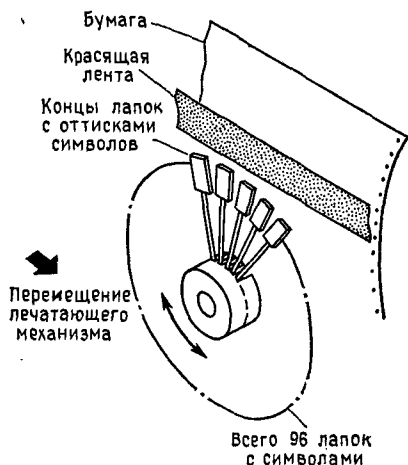


Рис. 22.5. Символы вытиснуты на «лепестках» печатающей головки типа «ромашка». В тот момент, когда при вращении колеса символ находится в позиции для удара, по лепестку ударяет молоточек.

модель 33 — это устаревшее и медленное по сегодняшним требованиям устройство, многие ее экземпляры все еще эксплуатируются.

Другая печатающая машинка — Selectric фирмы IBM — имеет печатающий узел, схожий с моделью 33, а именно шаровую печатающую головку (обычно называемую гольфбол — мячик для гольфа) с оттиснутыми на ее поверхности символами. По получении соответствующего кода головка автоматически поворачивается вокруг вертикальной оси, делает нужный

поворот по горизонтали и подогнанным таким образом символом ударяет по красящей ленте и бумаге. Как и в случае цилиндра модели 33 Teletype, печатающая головка пишущей машинки Selectric имеет 2 степени свободы (рис. 22.4). Для подключения этой машинки к мини- и микроЭВМ промышленностью выпускается целый ряд адаптеров.

Как модель 33, так и машинка Selectric в момент своего создания не предназначались для ЭВМ и стали использоваться с ними в качестве печатающих устройств только потом. В этом смысле от них отличается группа устройств, в которых символы целиком выбиты по краям специальной крутящейся печатающей головки. Существуют два варианта такой головки: типа **ромашки** и типа **втулки**; принцип работы их одинаков.

Печатающая головка типа «ромашка» (рис. 22.5) представляет собой крутящийся диск с гибкими лапками-«лепестками» (название головки вызвано именно ее внешним сходством с цветком). На каждом лепестке вытиснуто зеркальное изображение какого-нибудь символа. Ромашка постоянно крутится, и в нужные моменты времени по лепесткам бьет (не крутящийся) молоточек. Оттиснутым символом лепесток бьет по красящей ленте, а та — по бумаге, оставляя отпечаток.

Печатающая втулка похожа на ромашку, только лепестки у нее загнуты вверх образца наподобие корзиночки. Символы находятся на гибких лепестках по краям «корзиночки». «Корзиночка» крутится вокруг своей оси и продвигается вдоль бумаги, причем, когда на позиции удара оказывается необходимый символ, по лепестку ударяет молоточек.

Как устройства, специально созданные для стыка с ЭВМ, печатающие устройства с крутящейся головкой имеют уникальные характеристики. Важнейшими такими характеристиками являются скорость печати (порядка 50 симв./с) и качество печати, приближающееся к тому, которое доступно на пишущих машинках. Это делает подобные печатающие устройства весьма ценным инструментом для систем символьной обработки.

Очень важно, что крутящиеся головки заменяемы: это позволяет использовать шрифты с пропорциональными межбуквенными пробелами, шрифты для чтения оптическими устройствами, головки с различными алфавитами или наборами специальных знаков (рис. 22.6).

Подобные устройства могут иметь широкую каретку, что позволяет вставлять в них бумагу шириной до 40 см; в некоторых устройствах используются два печатающих узла, совместно выпечатавающие каждую строку (рис. 22.7). Различные типы ромашковых печатающих устройств показаны на рис. 22.8.

Работу посимвольных печатающих устройств обычно принято представлять как последовательную распечатку сначала

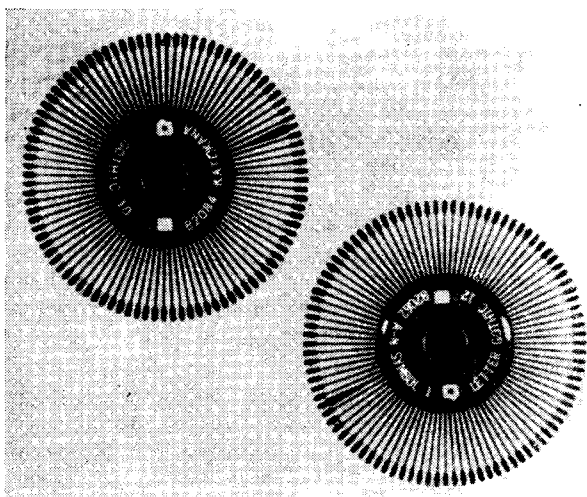


Рис. 22.6. Печатающие ромашки взаимозаменяемы, что позволяет работать с различными шрифтами (фирма Qume/ITT).

одной строки от начала и до конца слева направо, затем возврат каретки, начало новой строки и так далее, т. е. как одностороннюю печать. В устройствах с крутящейся головкой возможна организация двунаправленной печати, при которой строка, следующая за фактически печатаемой строкой сперва запоминается в буферной памяти, встроенной в схему управления устройством, а потом выводится на печать; при этом возможна выдача в обоих направлениях. Критерием выбора направления служит минимизация времени печатания. Если новая строка длинна, а печатающая головка уже находится в самой правой позиции, то эта строка будет выведена движением головки справа налево, навстречу тому, как она будет читаться. Тогда можно будет избежать потери времени на возврат головки через всю ширину бумаги в самую левую позицию. В схеме управления устройством подсчитываются времена, необходимые для распечатки строки как в одном, так и в другом направлении, сравниваются между собой и принимается наиболее экономное решение.

В некоторых устройствах можно программировать выдачу графического материала, предусматривая сдвиг головки на неполное межбуквенное расстояние по горизонтали и сдвиг бумаги на неполное межстрочное расстояние по вертикали. На рис. 22.9 показаны полученные таким образом кривые и примеры используемых символов. Если, например, номинальное межбуквенное расстояние соответствует 10 символам на 25,4 мм,

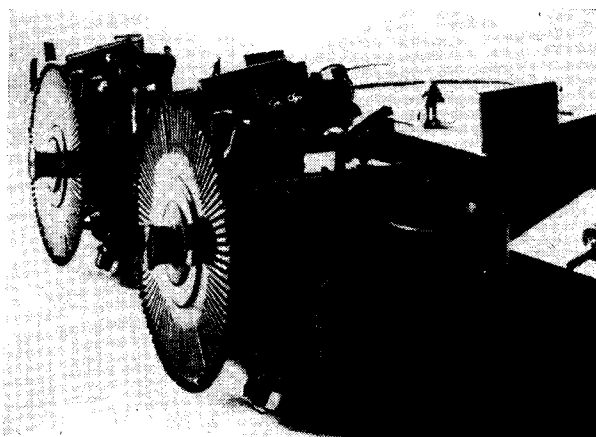


Рис. 22.7. Сдвоенный печатающий узел типа «ромашка» позволяет вдвое увеличить скорость работы (фирма Qume/ITT).

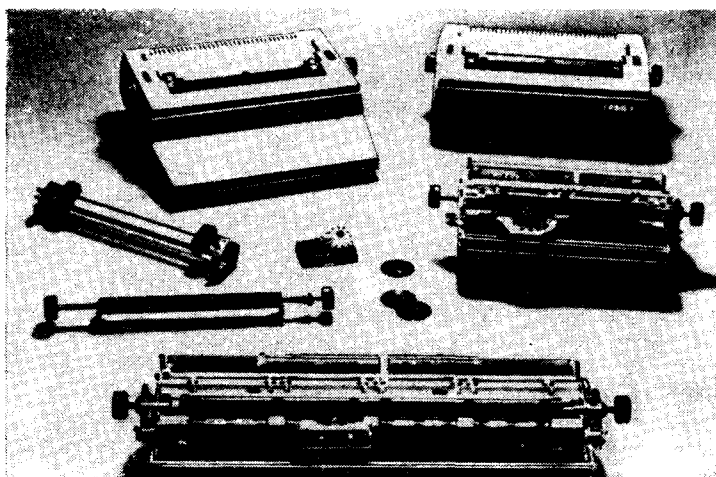
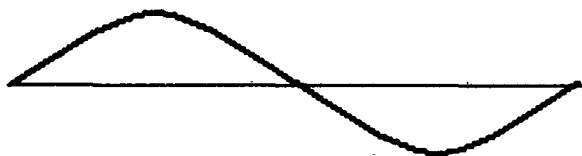


Рис. 22.8. Некоторые модели ударных печатающих устройств типа «ромашка» и компоненты к ним (фирма Qume/ITT).



ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789_.,%&@:'"~[]

abcdefghijklmnopqrstuvwxyz >*<./ç\{\$(\&)=--;#[+~^

Рис. 22.9. Пример твердой копии, выполненной устройством типа «печатающая втулка» с учетом его программируемых возможностей (NEC Information Systems).

Устройство типа «печатающая втулка» обладает следующими возможностями: подключение по интерфейсу RS232, токовой петле, к входной шине процессора 8080 или по какому-нибудь другому интерфейсу, описание которого нужно сообщить изготовителю; обеспечивается совместимость с печатающими устройствами фирм Diablo, Qume и Septipis. Имеются возможности программируемой вертикальной табуляции, двунаправленной печати, пропорциональных межбуквенных пропусков, программируемой горизонтальной табуляции, печатания двухцветной красно-черной лентой. Допускается печать надстрочных и подстрочных индексов, а также графических изображений. Каждая из применяемых печатающих втулок содержит не менее 100 пропорционально печатаемых символов. Пример набора символов на одной из втулок приведен в двух последних строках распечатки.

то программно-управляемое смещение по горизонтали может осуществляться, скажем на 0,2 мм.

Аналогично, если вертикальный прогон бумаги (осуществляемый командами *перевод строки* или *смещение*¹⁾), производится с шагом, соответствующим 6 строкам на 25,4 мм, может быть реализована возможность программного смещения по вертикали на 0,5 мм. Смещая, таким образом, печатающую головку по горизонтали, а бумагу — по вертикали (вниз или вверх), можно выполнить вывод любой графической и текстовой информации.

22.5. ПОСТРОЧНЫЕ УДАРНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА С ЦЕЛЬНЫМ ОЧЕРТАНИЕМ СИМВОЛОВ

Обычно устройства этой группы просто называют строчными печатающими устройствами; в них все символы и пробелы, составляющие строку (как правило, длиной до 132 символов), выпечатываются одновременно для каждой строки. После такой выдачи происходит прогон бумаги и печать следующей строки. Скорость работы строчных печатающих устройств колеблется от сотен до тысяч строк в минуту.

¹⁾ Команда смещение (form feed) управляет перемещением позиции печати в заранее назначенную точку печатающего устройства, например, на следующую страницу, фрагмент и т. п. — *Прим. ред.*

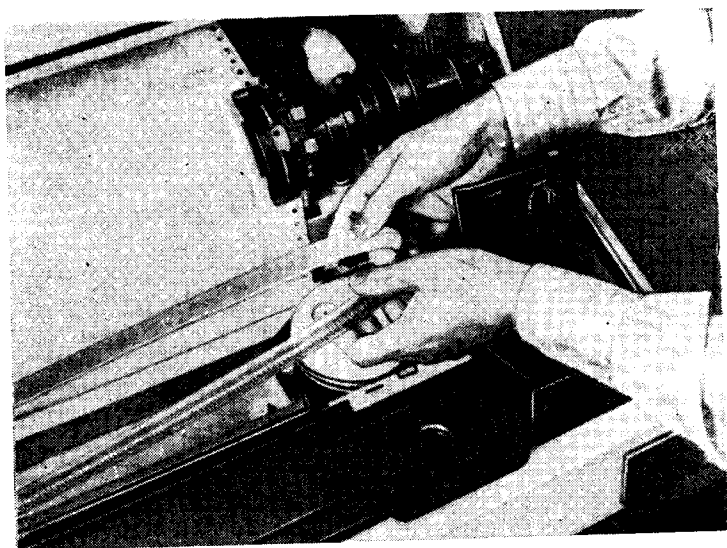


Рис. 22.10. Шрифтовая лента в этом ленточном печатающем устройстве легко заменяема, что делает возможной работу с разными наборами символов (фирма Dataproducts).

Высокая скорость и стоимость (по сравнению с посимвольными печатающими устройствами) строчных устройств определяет сферу их применения: они используются при распечатке больших объемов информации и значительно реже — при работе с микроЭВМ. На этих устройствах распечатывают, например, выписки из счетов дебиторов, длинные платежные ведомости, списки номенклатуры в ассортименте и другие объемные выдачи коммерческих расчетов.

Для подвода каждого печатающего символа в нужное место строки используется шрифтоноситель двух типов: цепь или лента, движущиеся горизонтально параллельно строке; или же барабан, состоящий, как правило, из 132 секций и вращающийся перед бумагой. В последнем случае символы, оттиснутые на каждой из секций барабана при его вращении проходят мимо бумаги и печатаемой строки в вертикальном направлении. Для обоих типов носителя, когда заданный символ подходит в нужную позицию, включается молоточек (один из 132 — по каждому на столбец) и через красящую ленту на бумагу наносится отпечаток.

Шрифтоноситель в виде ленты для ударного строчного устройства показан на рис. 22.10. На ленте оттиснуты символы для печати; сама она протянута перед бумагой и вращается от двух

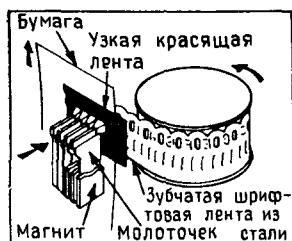


Рис. 22.11. В ленточных печатающих устройствах (называемых еще звонковыми) набор символов оттиснут на шрифтовой ленте (фирма Mannesmann Tally).

Молоточки, по одному в каждой печатной позиции символа, в нужные моменты ударяют по ленте.

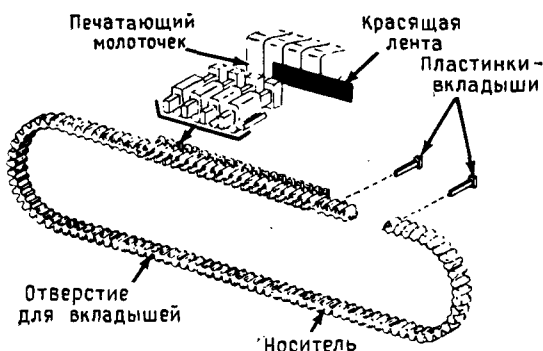


Рис. 22.12. В цепных печатающих устройствах символы оттиснуты на пластинках, вращающихся вместе с цепью (фирма Teletype).

приводных роликов. Между шрифтовой лентой и бумагой располагают еще одну ленту, красящую. При движении шрифтовой ленты мимо каждой из 132 горизонтальных позиций по ней в соответствующие моменты времени ударяет один из молоточков, расположенных за бумагой, производя таким образом в нужной позиции отпечаток нужного символа (рис. 22.11). Специальная схема отслеживает, когда и в какой позиции оказывается каждая буква или цифра, и включает по мере необходимости молоточек. По тому же принципу работает и цепное печатающее устройство (рис. 22.12). Здесь, однако, вдоль бумаги и красящей ленты вращается специальная цепь. В каждое звено цепи вставлена пластинка с оттиснутым на ней символом. Молоточки находятся за бумагой против каждой из 132 позиций и ударяют по проходящим мимо символам в нужные моменты времени.

Шрифтовая цепь или лента обычно содержит не один, а несколько наборов символов. Поскольку чем меньше символов в наборе, тем больше скорость печатания строк, то во многих строчных печатающих устройствах использован только 64-символьный набор ASCII, в который включены только заглавные буквы. В некоторых устройствах используется более простой набор, состоящий из 26 символов алфавита, 10 цифр и нескольких специальных знаков. Во многих устройствах упрощена процедура смены шрифтоносителя: это сделано для того, чтобы дать возможность пользователям быстро заменять набор символов и в конечном счете повысить эффективность устройства.

Принцип работы барабанного печатающего устройства проиллюстрирован на рис. 22.13. Все 64 (или 96) символов набора

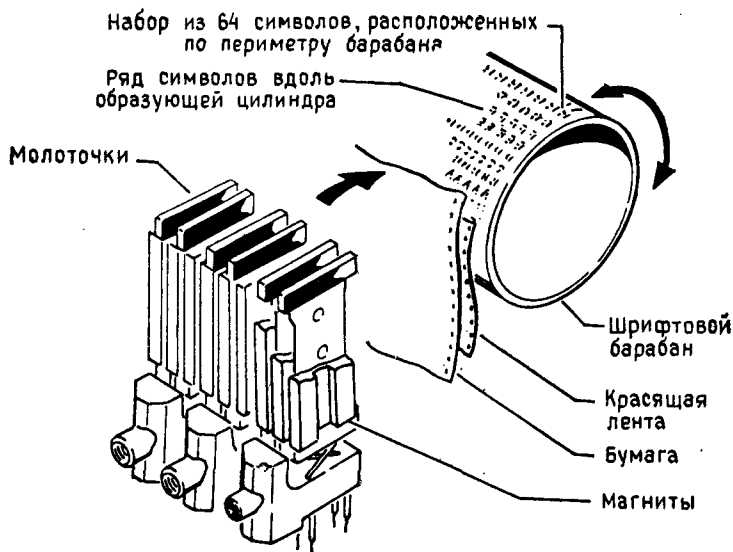


Рис. 22.13. В барабанных печатающих устройствах символы выбиты по поверхности барабана так, что каждой позиции в строке соответствует полный набор (фирма Dataproducts).

Молоточек ударяет по бумаге всякий раз, когда нужный символ оказывается против соответствующей позиции в строке.

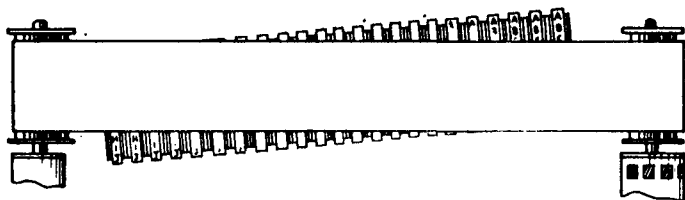


Рис. 22.14. Наклон красящей ленты по отношению к барабану позволяет регулировать износ ленты (фирма Control Data).

выбиты на каждой из 132 дорожек барабана. Спереди барабана находится бумага, а за ней красящая лента (рис. 22.14). Печатающие молоточки бьют по своим дорожкам, и через красящую ленту отпечаток наносится на бумагу.

22.6. ПОСТРОЧНЫЕ БЕЗУДАРНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА С ЦЕЛЬНЫМ ОЧЕРТАНИЕМ СИМВОЛОВ

Этот тип устройств используется в фотонаборных системах для быстрого построчного воспроизведения текста. Символы в них формируются на основе фото-, ксерографической или лазер-

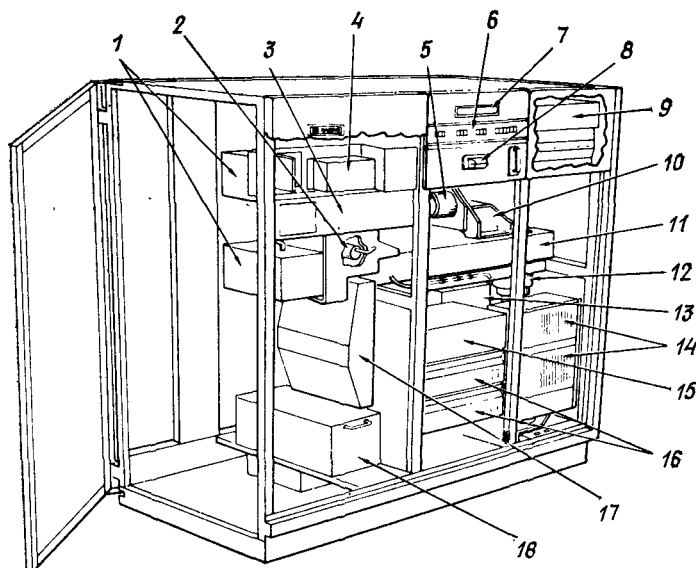


Рис. 22.15. Внутреннее устройство фотооптического печатающего строчного устройства, работающего под управлением мини-ЭВМ и используемого для фотонабора (фирма Harris).

1 — кассеты; 2 — резатель фотобумаги; 3 — внутренняя панель управления; 4 — управление подачи; 5 — формирующие линзы; 6 — наружная панель управления; 7 — цифровой индикатор; 8 — устройство считывания перфоленты; 9 — встроенная ЭВМ; 10 — проецирующее зеркало; 11 — станция оптической системы; 12 — ЭЛТ; 13 — электронный блок управления ЭЛТ; 14 — стойки с печатными платами; 15 — контроллер дисков; 16 — дисковые накопители с закодированными очертаниями символов; 17 — устройство подачи фотобумаги; 18 — коробка автоматического сборника.

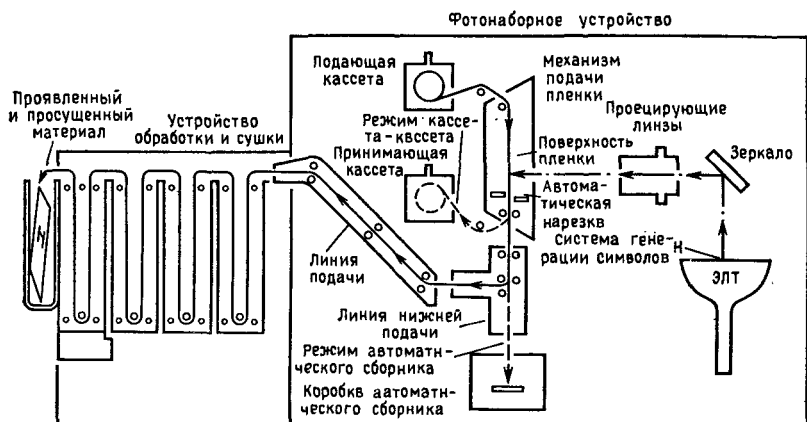


Рис. 22.16. Схема организации фотокопировального строчного печатающего устройства (фирма Harris).

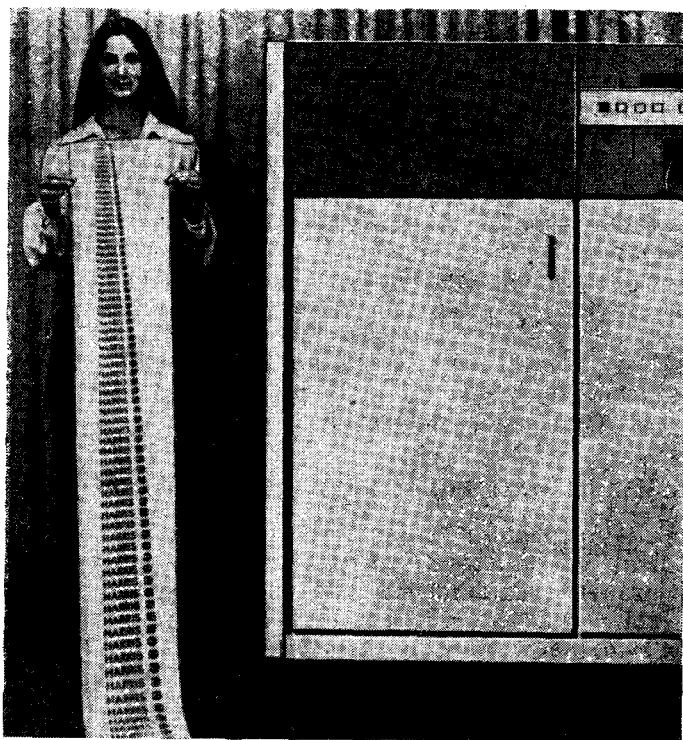


Рис. 22.17. Образец выдачи фотооптического строчного печатающего устройства (фирма Harris).

ной технологии. Это специализированные устройства; они дорогостоящи и поэтому очень редко используются в качестве внешних для мини- и микроЭВМ. Более того, они сами включают в состав мини- или микроЭВМ в качестве устройств управления.

Исключительно высокую скорость печати имеют **лазерные печатающие устройства**, сочетающие лазерную и ксерографическую технологии печатания: к 1980 г. скорость печати составляла до 18 000 или 21 000 стр./мин (300 или 350 стр./с). В лазерном печатающем устройстве луч лазера под управлением ЭВМ вырисовывает текст на электрически заряженном вращающемся барабане или же на специальной ленте. Частишки красителя прилипают к заряженным участкам, затем все изображение переносится на бумагу. Процесс завершается подогревом бумаги, в результате чего красящие частички спекаются вместе.

Управление **фотонаборными установками** с помощью мини- и микроЭВМ реализуется следующим способом. Выбранные вы-

числительным устройством образцы символов в соответствии с требуемым размером и очертанием проектируются оптической системой на фоточувствительную бумагу (или на фотопленку); таким образом набираются все строки копируемого документа. Далее бумага или пленка проявляется и используется для изготовления стереотипа для книг, журналов и газет.

Формирование символов осуществляется либо с помощью электронно-лучевой трубки и генератора символов (их очертания хранятся в памяти машины), либо чисто оптическим способом с использованием масок символов, с управлением от мини-ЭВМ. Устройство, схема организации и внешний вид фотонаборной установки на базе электронно-лучевой трубки и управляемой миниЭВМ показаны на рис. 22.15—22.17. Такие установки являются хорошим примером безударных построчных устройств специального назначения.

22.7. ТОЧЕЧНО-МАТРИЧНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА

Точечно-матричные символы — это символы, отпечатанные отдельными точками так, что группа точек воспроизводит нужное очертание буквы, цифры или иного символа. Такой метод печати широко используется в печатающих устройствах, применяемых в составе с мини- и микроЭВМ.

Точки могут выпечатываться как ударным, так и безударным способами, а устройства могут быть и посимвольными, и построчными. Различные применяемые технологии печати — ударная, термическая, электрочувствительная и струйная — будут рассмотрены ниже.

На рис. 22.18 приведен пример того, как можно отпечатать букву А, используя в одном случае матрицу 5 точек в ширину и 7 в высоту (5×7) и в другом случае матрицу 9×7 , в кото-

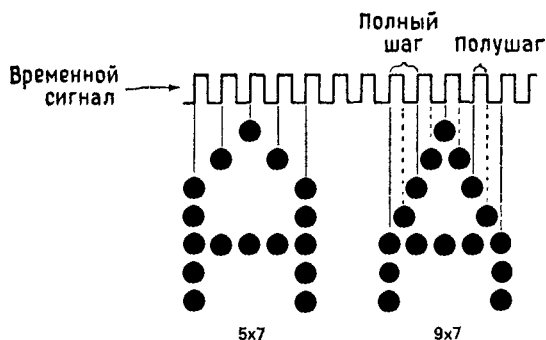


Рис. 22.18. Вид буквы А в представлении на двух матрицах разного размера (С. Itoh Electronics).

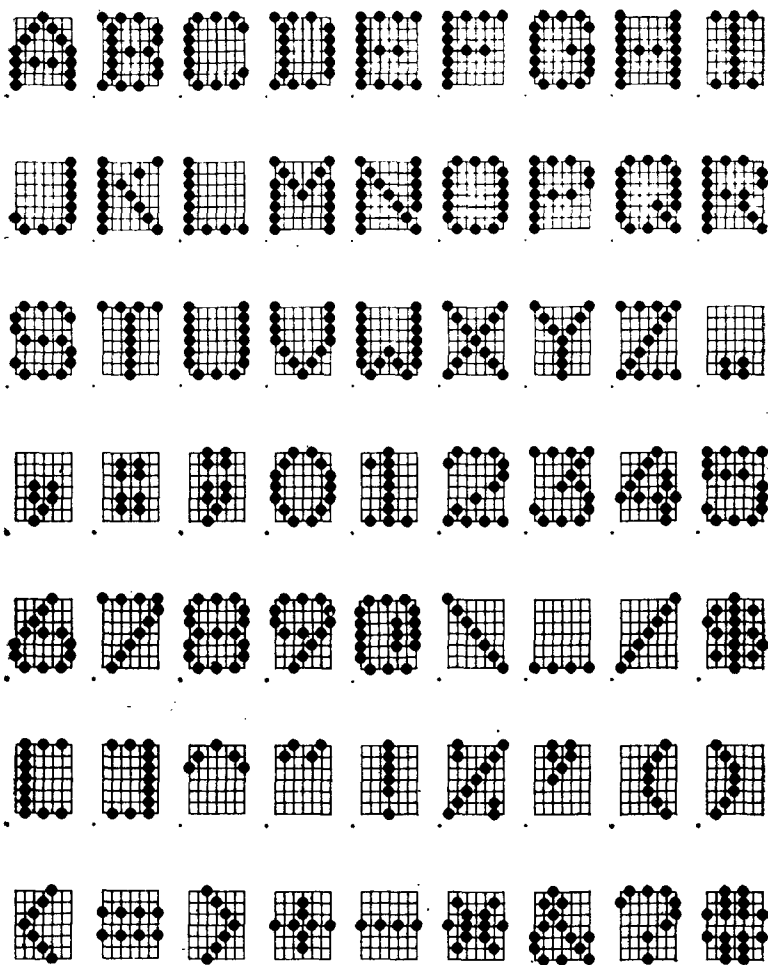


Рис. 22.19. 63-символьный набор, представленный на матрице 7×7 ; отсутствует пробел (фирма Gentronics Data Computer).

рой расстояние между точками по горизонтали составляет полшага. Другие применяемые типы матриц — это 7×7 , 8×8 , 7×9 , 9×9 и 9×12 . Полный 63-символьный печатный набор, полученный с помощью матрицы 7×7 , показан на рис. 22.19. На рис. 22.20, *a, б* приведен формат нормального и увеличенного символа, а на рис. 22.21 — текст, отпечатанный нормальным, компактным и двумя увеличенными форматами символов. Из этих рисунков видно, что матричные печатающие устройства обеспечивают широкий набор и разнообразие символов.

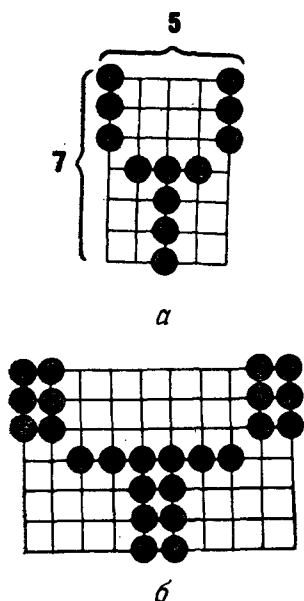


Рис. 22.20. Способ печати букв увеличенного размера на примере буквы «Y» (фирма Gentronics Data Computer).

а — нормальный формат; б — увеличенный формат.

В том случае, когда необходимы только заглавные буквы алфавита, часто используют матрицу 5×7 . Однако эта матрица не позволяет приемлемо представить все строчные буквы, поскольку у последних встречаются элементы, заходящие за нижнюю границу строки. Для набора, включающего и заглавный, и строчный алфавиты, используют матрицу 7×9 , с помощью которой можно реализовать не только все строчные буквы, но и выполнить подчеркивание символов.

В точечно-матричных устройствах можно напечатать любую комбинацию точек в соответствии с имеющимися в матрице позициями. Каждый из 128 кодов ASCII, поступающих на устройство, подается на вход микросхемы постоянного запоминающего устройства (ПЗУ), которая и формирует нужную конфигурацию точек. Таким образом, заменив в печатающем устройстве микросхему ПЗУ, можно перестроиться на печать любого другого алфавита или набора графических символов.

В некоторых матричных печатающих устройствах соседние точки могут частично перекрываться как по горизонтали, так и по вертикали; в таких устройствах иногда образуют блоки по 4 или более соседних точек и используют эти блоки для выде-

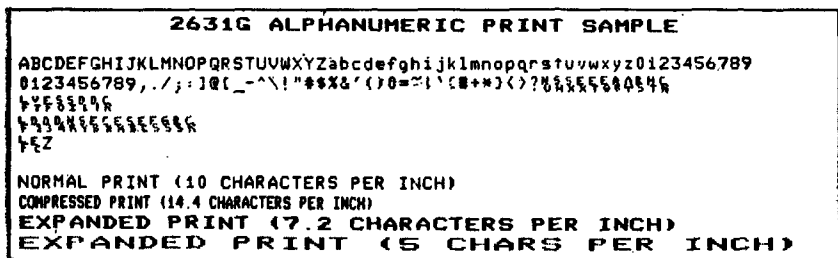


Рис. 22.21. Образцы печати с помощью точечной матрицы заглавных и строчных букв алфавита. Показаны нормальный (10 симв./дюйм), компактный (14,4 симв./дюйм) и увеличенные (7,2 и 5 симв./дюйм) форматы шрифта (1 дюйм = 25,4 мм) (фирма Hewlett-Packard).

USASCII	A B C D E F G H I J K L M N O P Q R S T U V W
ARABIC	٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ ١٠ ١١ ١٢ ١٣ ١٤ ١٥ ١٦ ١٧ ١٨ ١٩ ٢٠ ٢١ ٢٢ ٢٣ ٢٤ ٢٥ ٢٦ ٢٧ ٢٨ ٢٩ ٣٠ ٣١ ٣٢ ٣٣ ٣٤ ٣٥ ٣٦ ٣٧ ٣٨ ٣٩ ٤٠ ٤١ ٤٢ ٤٣ ٤٤ ٤٥ ٤٦ ٤٧ ٤٨ ٤٩ ٥٠ ٥١ ٥٢ ٥٣ ٥٤ ٥٥ ٥٦ ٥٧ ٥٨ ٥٩ ٦٠ ٦١ ٦٢ ٦٣ ٦٤ ٦٥ ٦٦ ٦٧ ٦٨ ٦٩ ٧٠ ٧١ ٧٢ ٧٣ ٧٤ ٧٥ ٧٦ ٧٧ ٧٨ ٧٩ ٨٠ ٨١ ٨٢ ٨٣ ٨٤ ٨٥ ٨٦ ٨٧ ٨٨ ٨٩ ٩٠ ٩١ ٩٢ ٩٣ ٩٤ ٩٥ ٩٦ ٩٧ ٩٨ ٩٩
CYRILLIC	а б в г д е ж з и й к л м н о п р с т у в ь
KATAKANA	チ ツ テ ナ ニ ヌ ネ ノ ハ ヒ フ ヘ ホ マ ミ ム メ ト ヲ ヨ
DRAW	

Рис. 22.22. Гибкость, которой обладает точно-матричная печать, позволяет выполнить распечатку не только букв иностранных алфавитов, но и графических символов (фирма Hewlett-Packard).

DATE: 7/17/78		HEWLETT-PACKARD COMPANY		ORDER #241402176	
SOLD TO: ABD LIMITED		SHIP TO: ABD LIMITED 123 E. 32 ST. HEWTOWN, MONTANA			
CUSTOMER ORDER NUMBER		CUSTOMER NO.		H.P. PURCH. NO.	
20329003451		115		CS471	
REQUIRE DATE		RATING		MFG. DISC.	
9-15-78		-		078	
				S.D. DISC.	
				047	
				TERMS	
				NET 30	
SHIP VIA INSTRUCTIONS: AIR BEST					
SPECIAL INSTRUCTIONS					
ITEM	PROD. NUMB.	DESCRIPTION		UNIT PRICE	QTY
01	2631A	PRINTER		3150	5
02	#240	2640 I/F		105	3
03	#210	1000 I/F		650	2
					1300
COMMENTS					
SHIP DATE		METHOD		CARRIER	
FREIGHT		C.O.D.		BOX NUMB.	

Рис. 22.23. Форма с занесенной в нее алфавитно-цифровой информацией, полученная на точечно-матричным печатающем устройстве (фирма Hewlett-Packard).

ления отдельных букв или значков. Возможности матричного печатающего устройства для воспроизведения ряда иностранных алфавитов проиллюстрированы на рис. 22.22. На рис. 22.23 показано, как на матричном печатающем устройстве можно получить распечатку выходной формы, состоящей из текстовой и графической информации.

22.8. ПОСИМВОЛЬНЫЕ УДАРНЫЕ МАТРИЧНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА

Обычно печатающая головка ударных матричных печатающих устройств состоит из набора проволоочек, которые ударяют

по бумаге через красящую ленту (рис. 22.24). Именно поэтому эту группу устройств иногда называют **проволочно-матричными печатающими устройствами**. Чаше всего печатающая головка состоит всего из одного столбца из 7 проволочек, хотя, вообще говоря, столбцов может быть 2 и больше (рис. 22.25). Внешний вид точечно-матричного печатающего устройства показан на рис. 22.26. Печатающая головка крепится на стержне, по которому может передвигаться горизонтально (рис. 22.27); печать происходит обычно в обоих направлениях. Иногда на одной каретке могут располагаться сразу две головки (рис. 22.28).

Рассмотрим для простоты головку, состоящую из 7 вертикально расположенных проволочек. Все 7 проволочек могут выталкиваться из головки (как правило, электромагнитами) в любой комбинации. Конкретную комбинацию указывает контроллер. Проволочки ударяют по ленте, а та — по бумаге, печатая один вертикальный столбец изображения выводимого символа. В случае устройства с одношаговой сеткой 5×7 головка далее продвигается на один шаг вперед, печатает второй столбец точек, и так до тех пор, пока не будут напечатаны все 5 столбцов.

Если используется полушаговая сетка, то процесс аналогичен, за исключением того, что при распечатывании символа происходит 9 горизонтальных продвижений головки (5 нормальных шагов, плюс 4 перемежающихся полшага), что фактически соответствует полушаговой сетке 9×7 .

Реально во всех ударных точечно-матричных устройствах печатающая головка состоит из 1 или 2 столбцов проволочек, а не из полного набора, скажем 5×7 . Это связано с тем, что с каждым новым столбцом увеличивается сложность механики и схемы управления головкой.

Из сказанного следует, что, строго говоря, точечно-матричные печатающие устройства печатают одновременно не полный символ, а лишь **один столбец** его изображения. Тем не менее скорость работы таких устройств достаточно высока (до 180 симв./с), что сравнимо или даже быстрее многих посимвольных устройств с цельным очертанием символов.

Большинство точечно-матричных печатающих устройств, кроме высокой скорости, обладают также способностью печатать в двух направлениях. Главный недостаток этого типа устройств заключается в том, что символы, которые они печатают, формируются из отдельных точек и не являются целостными. Это ограничивает их применение для задач символьной обработки, но сохраняет их эффективность благодаря простоте механической конструкции для других прикладных областей. В настоящее время ведутся работы над созданием таких точечно-матричных устройств, ориентированных на массового поль-

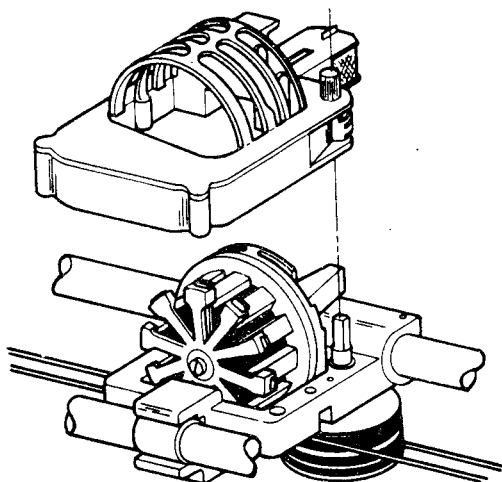


Рис. 22.27. Перемещение точечно-матричной печатающей головки вдоль каретки осуществляется с использованием двух опорных стержней (фирма Control Data)

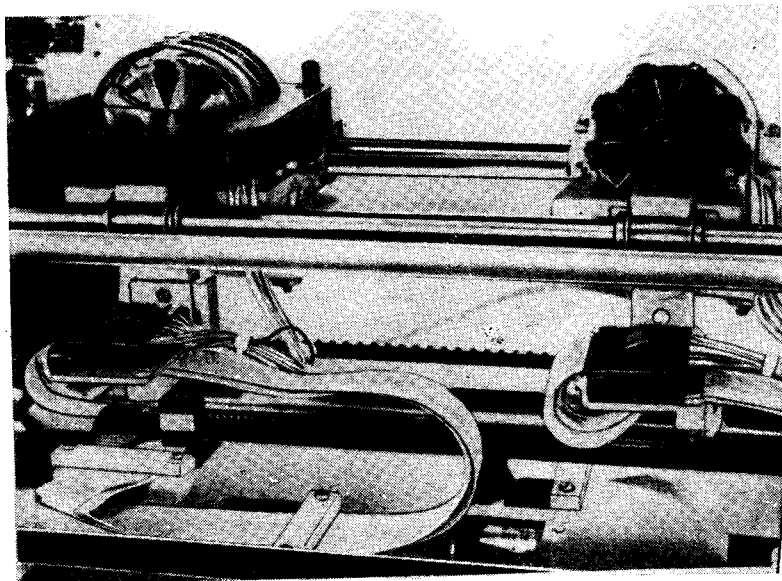


Рис. 22.28. Для увеличения скорости печати вдвое на этой каретке размещены две точечно-матричные печатающие головки (фирма Control Data).

зователя, которые бы печатали мелкими точками на сетке большого размера. Подобное сочетание позволило бы сделать точечно-матричные символы практически неотличимыми от целостных.

22.9. ПОСТРОЧНЫЕ УДАРНЫЕ ТОЧЕЧНО-МАТРИЧНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА

В таких устройствах используется горизонтальная печатающая гребенка (рис. 22.29) шириной в лист бумаги. Гребенка сделана из цельной металлической полоски и имеет 132 «зуба»-молоточка, на каждом из которых расположен небольшой выступающий шип. В состав устройства входят электромагниты, которые, получая сигналы от генератора символов, притягивают молоточки к себе и отпускают их. При отпускании шип через красящую ленту ударяет по бумаге, печатая точку (рис. 22.30). Каждый из молоточков все время находится в зоне печати одного из символов, а расстояние между двумя соседними молоточками составляет один пробел. Гребенка совершает колебательные движения по горизонтали, что позволяет ей распечатать любую из перекрывающихся 5 точек, из которых складывается один горизонтальный ряд в изображении каждого символа. Печать следующего горизонтального ряда всех символов в строке происходит после смещения бумаги на одну точку вниз по вертикали.

Таким образом, устройство печатает сначала первый ряд точек для каждого из 132 символов, затем второй и т. д. до седьмого, пока не будут сформированы все 132 матрицы 5×7 . Хотя на бумаге одновременно выпечатывается один ряд точек, по существу такое устройство является построчным печатающим устройством с эффективной скоростью порядка 300 строк в минуту.

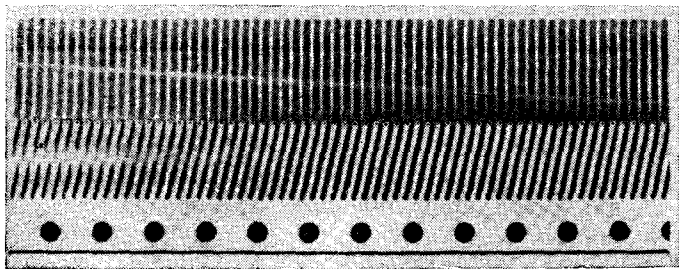


Рис. 22.29. Печатающая гребенка точечно-матричного строчного печатающего устройства (фирма Mappesmann Tally).

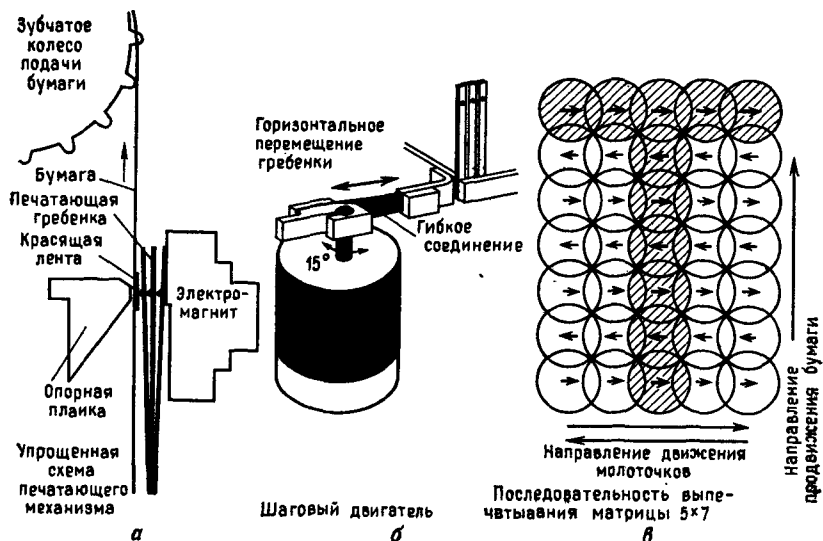


Рис. 22.30.

а — зубцы печатающей гребенки, расположенные каждый в одной из горизонтальных позиций печати символа, оттягиваются от бумаги электромагнитами; б — движение шипов на зубцах гребенки вдоль каждой печатной позиции осуществляется с помощью шагового электродвигателя, соединенного с гребенкой гибкой пластинкой; в — порядок выпечки точек гребенкой (фирма Mappesmann Tally).

22.10. БЕЗУДАРНЫЕ ТОЧЕЧНО-МАТРИЧНЫЕ ПЕЧАТАЮЩИЕ УСТРОЙСТВА

В безударных точечно-матричных печатающих устройствах отпечаток получается без соприкосновения печатающей головки с бумагой. Речь идет, следовательно, во-первых, о бесшумных устройствах, а во-вторых, о таких, на которых нельзя получить копию документа через копировальную бумагу, поскольку отсутствует сила, «продавливающая» символ через слой из обычной и копировальной бумаги. Тем не менее безударные устройства весьма полезны при получении твердой копии в единичных экземплярах, при распечатке результатов расчета на калькуляторе, при получении копий с экрана дисплея, регистрации данных в промышленных установках и во многих других случаях.

В современных безударных точечно-матричных печатающих устройствах используется одна из следующих технологий получения отпечатка: термическая, электрочувствительная, электростатическая, струйная.

Термические печатающие устройства

В термических печатающих устройствах применяется разогрев электрическим током в позициях точечной матрицы, соответствующих очертанию символа. В результате разогрева происходит окрашивание специального теплочувствительного слоя, которым покрыта бумага, в черный или голубой цвет. Головки в термическом печатающем устройстве либо движутся горизонтально, печатая, как и в ударных точечно-матричных устройствах, столбец за столбцом поточечное изображение каждого символа, либо неподвижны и выпечатывают сразу одну горизонтальную строку точек.

На рис. 22.31 показан вид головки для последовательной печати символов с матрицей высотой в 7 точек. В верхнем правом углу головки расположены нагревательные элементы — обычные сопротивления. По электрической схеме на сопротивления подается ток, они разогреваются и осуществляют местный подогрев бумаги в точках соответствующего столбца в изображении символа; в результате подогрева бумага окрашивается.

В построчных термических устройствах в качестве головки используется керамическая планка, на которую нанесен точечный ряд нагревательных элементов. Термические печатающие устройства применяются обычно в калькуляторах, в различных приборах, для записи текущего состояния экрана дисплея (рис. 22.32, 22.33).

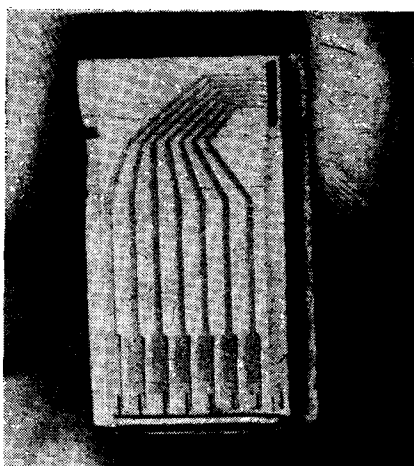


Рис. 22.31. Термическая точечно-матричная печатающая головка для посимвольной печати (фирма Dataproducts).

Электрочувствительные печатающие устройства

В этих устройствах ток, проходя через кончик иглы или шип печатающей головки, вызывает электрическую дугу, которая взаимодействует со специальным покрытием, нанесенным на бумагу и чувствительным к изменению напряжения. Во многих электрочувствительных печатающих устройствах в качестве



Рис. 22.32. Термическое построчно печатающее устройство программируемого калькулятора.

Неподвижная печатающая головка, состоящая из выстроенных в ряд нагревательных элементов, позволяет выводить строки по 20 символов.

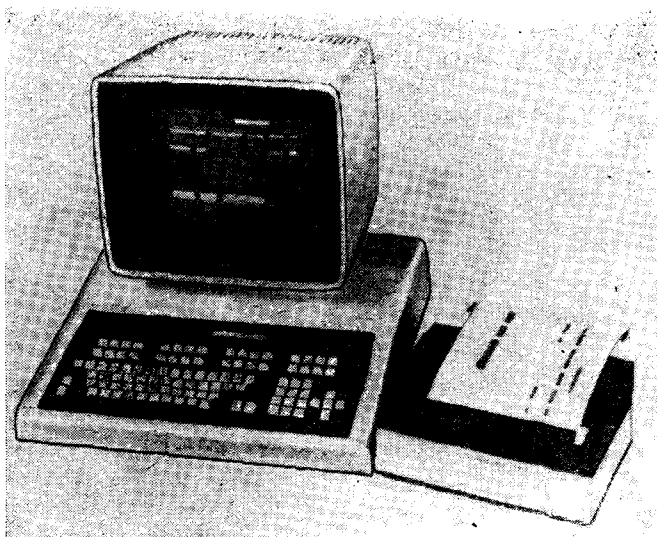


Рис. 22.33. Термическое печатающее устройство, используемое для записи содержимого экрана дисплея.

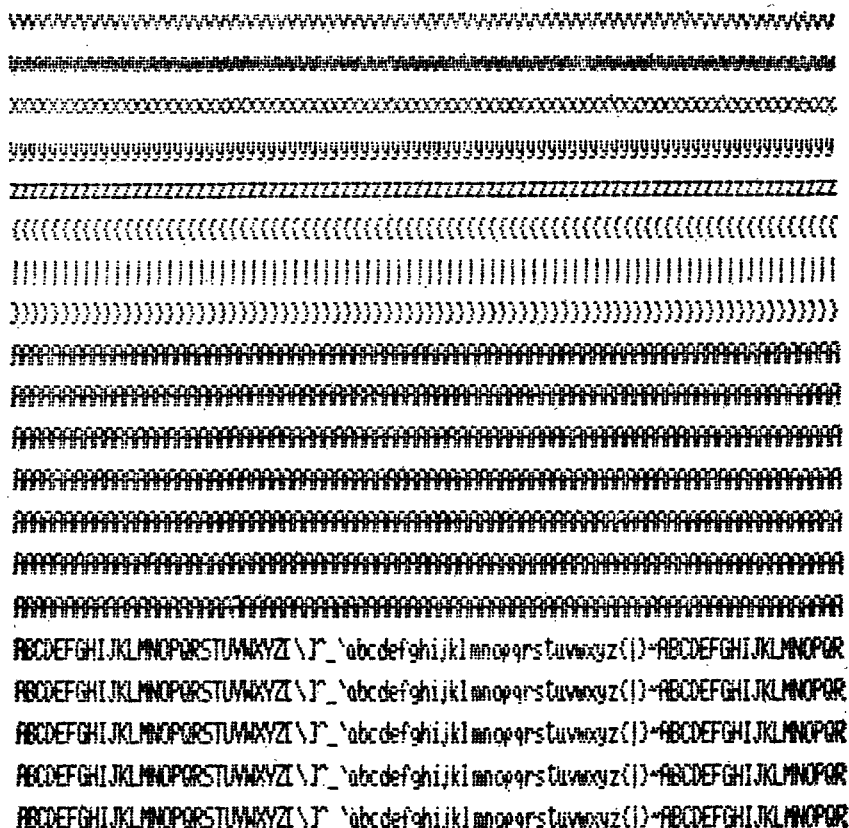


Рис. 22.34. Пример копии оттиска в натуральную величину, полученного на электрочувствительном печатающем устройстве.

Такие устройства часто используются для записи содержимого с экрана. Бумага покрыта тонким слоем алюминия.

покрытия бумаги используется алюминий, который при подаче напряжения к печатающей головке испаряется и в этом месте образуется черная точка. В отличие от термических печатающих устройств, в которых выдача блекнет под воздействием тепла или солнечного света, на бумагу с алюминиевым покрытием не действует ни температура, ни свет, ни вода, т. е. в этом случае можно говорить об относительно долговечной форме документирования (рис. 22.34). Электрочувствительные печатающие устройства часто используются для записи изображения с экрана дисплея.

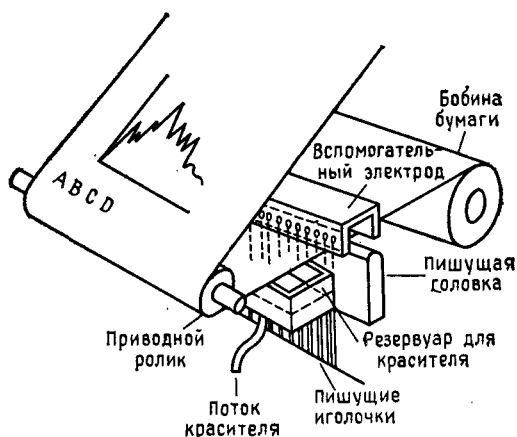


Рис. 22.35. Схема электростатической печати.

На бумагу, покрытую слоем диэлектрика, наносится электрический заряд. Группы заряженных точек, соответствующие очертаниям символов, притягивают на себя частички черного красителя.

Электростатические печатающие устройства

Принцип действия этих устройств состоит в том, что на иглолочки, используемые для формирования точно-матричного изображения символов подается напряжение, которое воздействует на бумагу со специальным диэлектрическим покрытием. По мере того как бумага, разматываясь из рулона, проходит мимо иглолочек, выстроенных в ряд, на ней остаются маленькие заряженные точки-следы. После этого бумага протягивается мимо черного красителя, частички которого прилипают к заряженным точкам, формируя образы символов (рис. 22.35).

Струйные печатающие устройства

В этих устройствах точно-матричные символы формируются в результате непосредственного выдувания капелек чернил на бумагу. В настоящее время используются две разновидности струйной технологии.

В **пусковых капельных системах** (рис. 22.36) капельки чернил под давлением выбрызгиваются из выпускных отверстий (форсунок), направленных в сторону бумаги. Сигналом для открытия каждого отверстия служит электрический импульс. Печатающая головка состоит из вертикального ряда выпускных отверстий, соотнесенного к столбцу точечной матрицы изображения символа. Эта головка передвигается горизонтально, выбрызгивая чернила из соответствующих форсунок и формируя шаг за шагом изображение символа.

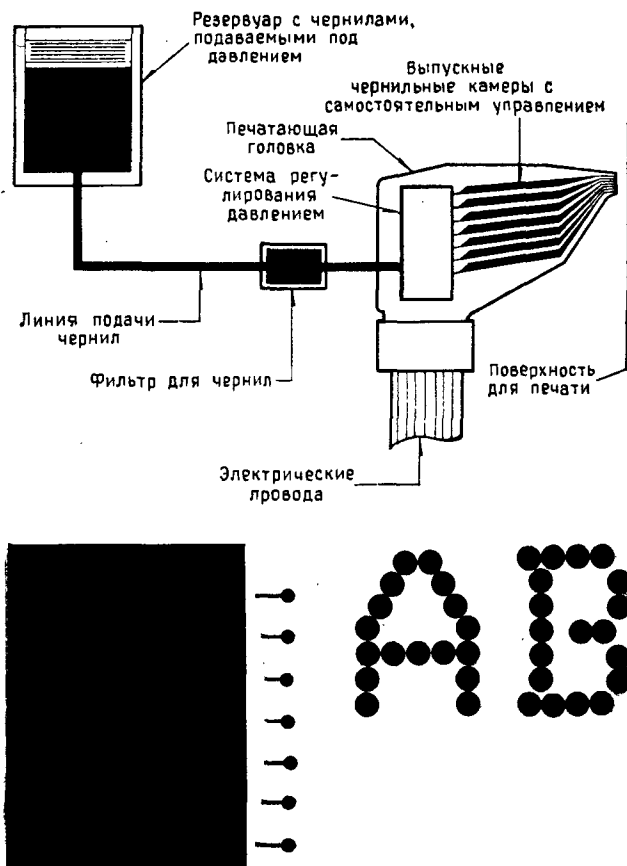


Рис. 22.36. Струйная печатающая головка формирует точно-матричные изображения символов, разбрызгивая струю из капелек чернил (Silonics).

В системах с **непрерывным потоком** чернила выдуваются из отверстий все время без перерывов. Поток капелек проходит через специальную трубку, где капельки приобретают электрический заряд (рис. 22.37). Далее поток проходит мимо пластин, напряжение на которых позволяет изменять направление его движения по горизонтали и по вертикали. Напряжение на отклоняющих пластинах регулируется электрической схемой генератора символов. Этот метод управления пучком (с помощью отклоняющих пластин) аналогичен используемому в электронно-лучевой трубке; здесь, однако, речь идет не о пучке электронов, а о потоке **чернильных капелек**. В те моменты, когда вырисовывать символ не нужно, чернильная струя отлавлива-

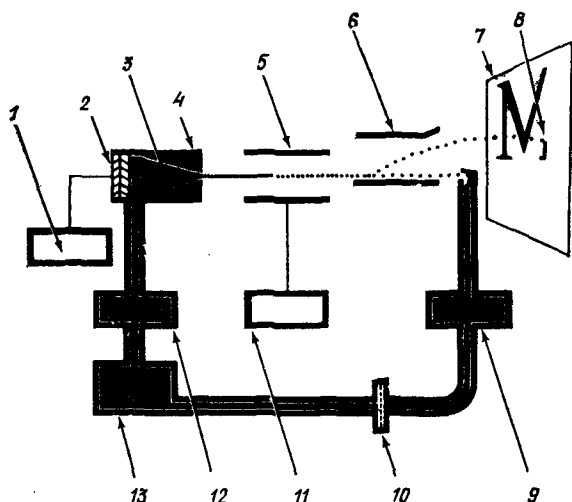


Рис. 22.37. В струйном печатающем устройстве с непрерывным потоком формирование символов происходит в результате отклонения заряженной чернильной струи отклоняющими пластинами аналогично тому, как направляется пучок электронов в электронно-лучевой трубке.

1 — привод пьезокристалла; 2 — пьезокристалл; 3 — генератор капель; 4 — форсунка; 5 — заряжающий электрод; 6 — отклоняющие пластины высокого напряжения; 7 — бумага; 8 — формирование символа; 9 — возврат неиспользованных чернил; 10 — фильтр; 11 — управление заряжающим электродом; 12 — насос; 13 — подача чернил.

ется и направляется в резервуар, из которого повторно поступает на разбрызгивание.

Строго говоря, в последней технологии нельзя говорить о точно-матричном изображении в том смысле, как мы это определили раньше, поскольку формирование символов осуществляется непрерывным потоком чернил. Тем не менее изображение складывается из отдельных капелек, хотя и очень малого размера, а значит является поточечным. По качеству текста струйные устройства с непрерывным потоком приближаются к устройствам, обеспечивающим цельное изображение символов. Фактически этот метод — гибрид обоих способов формирования изображения. Струйные устройства в отличие от безударных устройств других типов не требуют для работы специальной бумаги.

22.11. ПОДВОД БУМАГИ

Для всех типов печатающих устройств возникает необходимость обеспечить подвод бумаги. Имеются три способа такого подвода: подвод бумаги силой трения, фиксированная зубчатая подача и приводная зубчатая подача.

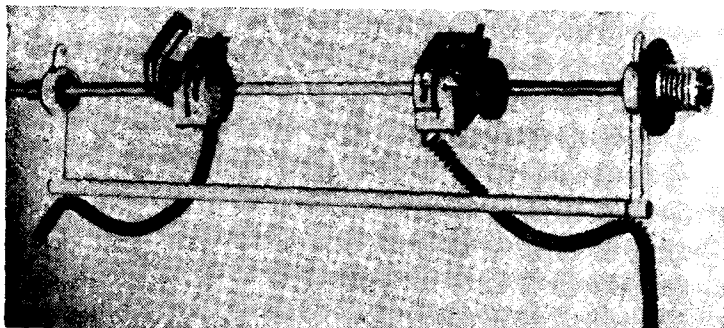


Рис. 22.38. Применение двух автономных зубчатых приводов позволяет обрабатывать печатные формы любой ширины (фирма Control Data).

Подвод силой трения — это способ, который применяется в обычных пишущих машинках. В устройстве имеется валик, вокруг которого загибается лист бумаги; по мере поворота валика бумага силой трения продвигается вперед. Такой способ подачи используется в сравнительно медленных устройствах.

При **фиксированной зубчатой подаче** используется бумага с отперфорированными по краям через равные промежутки отверстиями. На корпусе устройства находится зубчатый привод; зубцы попадают в отверстия, и, проворачиваясь, привод прокручивает бумагу вперед. Этот способ применяется в печатающих устройствах средней производительности и необходим в случае длинных «лент» бумаги (поступающей из рулона или сложенной «гармошкой» стопки), поскольку не приводит к накоплению сдвигов в больших распечатках. Зубчатый привод здесь жестко закреплен на корпусе и не допускает регулировки по горизонтали. Последнее обстоятельство заставляет использовать для каждого устройства бумагу только определенной ширины.

Приводная зубчатая подача состоит из двух автономных зубчатых приводов (рис. 22.38). Расстояние между приводами регулируется, что позволяет использовать бумагу разной ширины. Имеется возможность программно управлять пропуском свободных мест в печатаемом тексте, увеличивая тем самым скорость работы устройства. Автономный зубчатый привод часто используется там, где требуется с высокой скоростью и в больших количествах обрабатывать разнообразные формы документов.

В некоторых устройствах имеется двойной набор приводов и разделенная несущая ось. Это позволяет одновременно обрабатывать сразу две формы с различными требованиями

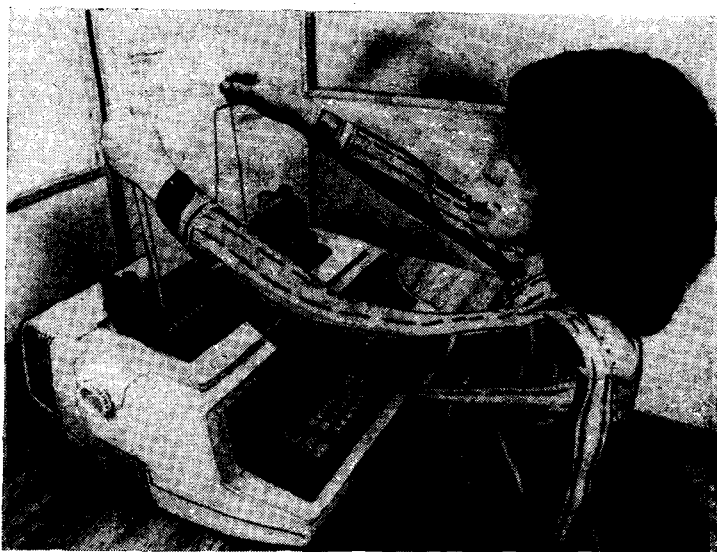


Рис. 22.39. Переднее расположение механизма подачи бумаги в устройствах печати для одновременной печати нескольких форм.

к ширине бумаги и интервалу между строками. Оба набора управляются самостоятельно и независимо друг от друга могут быть установлены на необходимый интервал. В таких печатающих устройствах можно, например, печатать в левой части имя владельца ценных бумаг и одновременно вести журнал учета таких бумаг в правой части. Кроме того, в правой части может вестись еще и диалог оператор — ЭВМ, тоже регистрируемый в журнале.

Формы на отдельных страницах можно, конечно, вставлять и продвигать вручную. Однако во многих устройствах реализована автоматическая вставка листов бумаги. Это позволяет придать письмам, отсылаемым адресатам, свой индивидуальный вид. Чистые листы бумаги при этом берутся обычно автоматически из стопки позади печатающего устройства.

В устройствах, рассчитанных на одновременное печатание нескольких форм, используется переднее расположение механизма подачи бумаги, аналогично применяемому в записях-счетах по аренде автомобилей (рис. 22.39). При этом оператор, работающий с устройством, должен опустить лист формы в наклонный пластиковый планшет, после чего необходимая подгонка по вертикали и пропуск свободных участков в форме осуществляется схемой управления устройством автоматически.

23

ГРАФОПОСТРОИТЕЛИ¹⁾

Л. Хоэнштейн

23.1. ВВЕДЕНИЕ

Графопостроителями называются устройства, которые позволяют зафиксировать результаты расчета на ЭВМ в виде твердой копии, состоящей из отрезков прямых и кривых линий. Последние являются теми элементами, из которых можно составить чертеж, схему, рисунок или вообще любое графическое изображение.

Данные для отрисовки на графопостроителе накапливаются или в оперативной, или во внешней памяти на магнитных лентах и дисках. Эти данные вызываются для отрисовки программой дисплейного процессора; команды этой программы интерпретируются дисплейным контроллером в сигналы, приводящие в движение рисующее перо. Настольный графопостроитель с плоской прямоугольной рисующей поверхностью, применяемый в качестве внешнего устройства мини- или микроЭВМ, показан на рис. 23.1. Существует много различных типов графопостроителей, однако их можно классифицировать по следующим признакам:

1. Метод формирования изображения: вычерчивание или сканирование.
2. Перемещение бумаги.
3. Размер устройства.
4. Назначение устройства — общее или специализированное.

23.2. МЕТОДЫ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ

Одним из методов формирования изображения в графопостроителях является вычерчивание этого изображения пишущим узлом, чаще всего пером (чернильным или типа «фломастер»). Графопостроители, использующие такой метод отрисовки, носят название **перьевых**. В перьевых графопостроителях имеется

¹⁾ Adapted from Computer Peripherals for Minicomputers, Microcomputers, and Personal Computers, by Louis Hohenstein. Copyright © 1980. Used by permission of McGraw-Hill, Inc. All rights reserved.

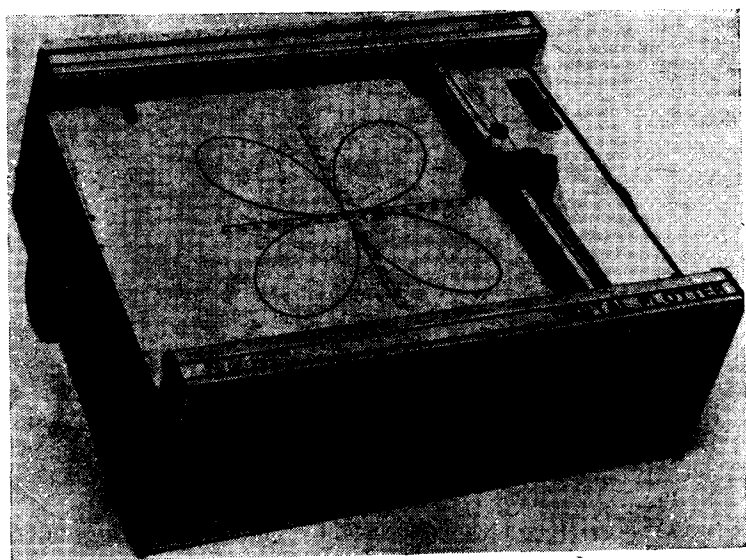


Рис. 23.1. Настольный перьевой планшетный графопостроитель (отделение Houston Instrument фирмы Bausch and Lomb).

один или более шаговых двигателей, которые перемещают перо, вычерчивающее маленькие прямолинейные отрезки длиной 0,25—0,032 мм. Множество таких отрезков вдоль координатных осей X и Y позволяет с очень хорошей точностью отрисовать кривую или отрезок прямой под любым наклоном. Графопостроители, на которых выполняется отрисовка в малых прямолинейных приращениях, называются **инкрементными**. У перьевых графопостроителей описанного типа имеются и другие названия — **XY-графопостроители** за их способность вычерчивания в двух координатных направлениях, или **векторные графопостроители** по аналогии с векторными дисплеями.

Если векторный графопостроитель одноперьевой, то он может рисовать только в одном цвете. Поэтому, когда требуется многоцветное изображение, программа должна быть разделена на сегменты, выполняющие отрисовку в определенном цвете. В начале (или в конце) каждого сегмента выполнение программы должно быть приостановлено, с тем чтобы дать возможность оператору заменить перо, затем возобновлено для вычерчивания нового сегмента.

В **многоперевых** графопостроителях (рис. 23.2) имеется возможность отрисовки многоцветных изображений (без остановки специально для смены цвета программы) или отрисовки

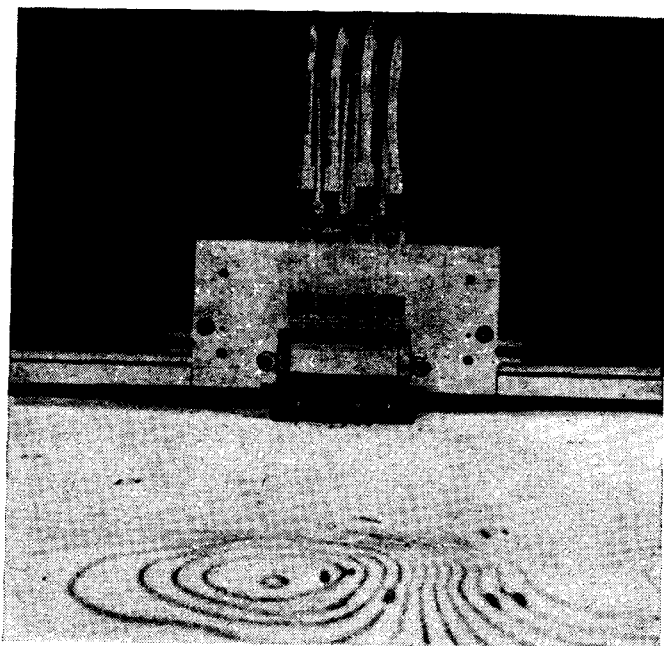


Рис. 23.2. В многоперьевых графопостроителях можно выполнять отрисовку линий разного цвета или разной толщины (например, линии уровня на топографических картах), не останавливая программу для смены пера (California Computer Products, Inc.).

линий разной толщины. Хотя перо и бумага являются традиционными атрибутами для графопостроителя, тем не менее возможно применение и других компонентов. Отрисовка может выполняться на матовой бумаге, кальке, прозрачной пленке для проецирования, фотопленке, стекле и даже на листе металла. Инструментами для отрисовки в специальных применениях могут служить, например, фоторисующая головка или термохромный гравировальный резец.

Если в векторных графопостроителях пишущий элемент непрерывно вычерчивает прямолинейные отрезки в любых направлениях, то в **растровых** устройствах требуется предварительный просмотр всего изображения и разбиение его на строки. Каждая полученная строка, составленная из отдельных точек (так называемый растр), выводится одна за другой электростатической печатающей головкой графопостроителя. Метод печати в электростатических графопостроителях основан на нанесении точечных зарядов на диэлектрическую бумагу с помощью

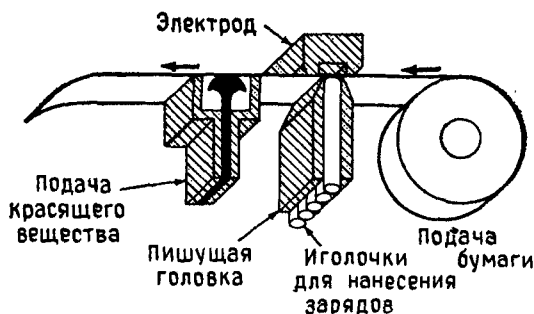


Рис. 23.3. Схема электростатического растрового графопостроителя (Versa-tes).

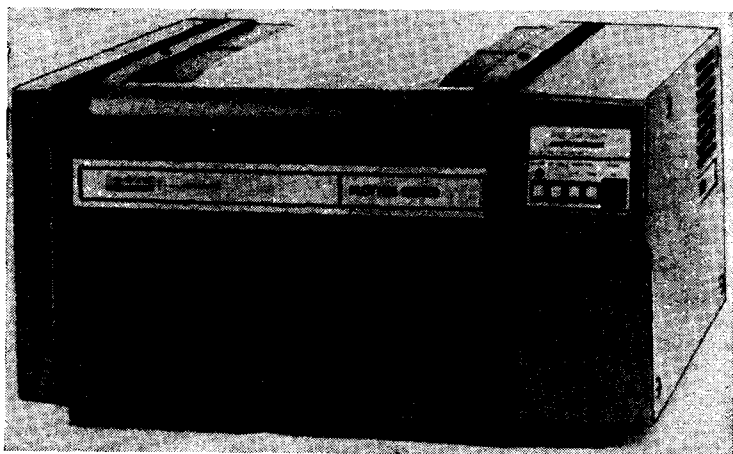


Рис. 23.4. Внешний вид растрового графопостроителя (отделение Houston Instrument фирмы Bausch and Lomb).

набора иголок (рис. 23.3). Место нанесения заряда должно соответствовать месту на рисунке, где требуется иметь обычную темную точку. Бумага проходит мимо красителя, вещество которого притягивается зарядами на бумаге. После этого бумажный лист высушивается вентилятором; высохшие частички красителя образуют желаемое изображение. Графопостроители такого типа «обеспечивают» размещение точек с плотностью от 4 до 8 точек на миллиметр и поэтому могут быть использованы как для вывода текста, так и произвольных графических изображений. Именно поэтому электростатические графопостроители часто называют печатающими графопостроителями (графическими печа-

тающими устройствами). Как видно из рис. 23.4, растровые электростатические графопостроители не имеют пишущего узла и внешне выглядят как алфавитно-цифровые печатающие устройства.

23.3. ПЕРЕМЕЩЕНИЕ БУМАГИ

Графопостроители можно разделить по способу перемещения бумажного носителя на два типа: в графопостроителях первого типа бумага неподвижна, в графопостроителях второго типа она перемещается. К первому типу относятся планшетные графопостроители, ко второму — рулонные и растровые графопостроители.

Планшетные графопостроители

Планшет графопостроителя подобен чертежной доске. Тонкий лист бумаги или другого материала для нанесения изображения размещается на планшете неподвижно. Отрисовка выполняется пишущим узлом, который может перемещаться в двух взаимно перпендикулярных направлениях. Размеры изображения ограничены размерами планшета. Поскольку вся поверхность планшета доступна для нанесения изображения, отрисовку отдельных сегментов можно выполнять в совершенно произвольном порядке.

В планшетных графопостроителях движение пишущего узла осуществляется вдоль штанги, концы которой перемещаются по двум параллельным направляющим. Направляющие проложены с обеих сторон планшета вдоль его длинных сторон. Каретка с пером (или с перьями) передвигается вдоль штанги. Одновременные перемещения штанги вдоль листа и каретки вдоль штанги обеспечивают перемещение пера по всему планшету.

Как штанга, так и каретка приводятся в движение шаговым электродвигателем. Управляющие команды задают малые перемещения пишущего узла вдоль координатных осей планшета. Команды на отрисовку графического изображения, поступающие от ЭВМ, в конечном счете и переводятся в эти перемещения.

Рулонные графопостроители

В то время как в планшетных графопостроителях бумага неподвижна, в рулонных графопостроителях она непрерывно движется. При этом бумага либо извлекается из стопки (рис. 23.5), либо разматывается от рулона (рис. 23.6). Отрисовка графического изображения выполняется по мере подачи

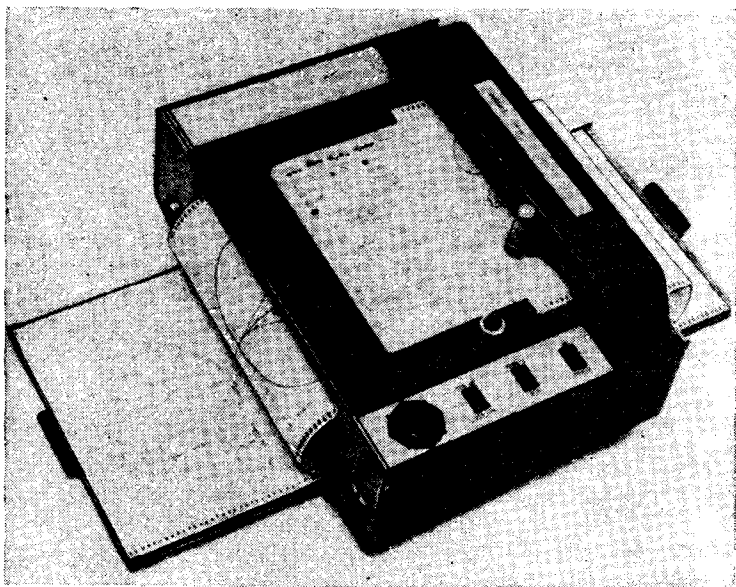


Рис. 23.5. Рулонный графопостроитель с извлечением бумаги из стопки (отделение Houston Instrument фирмы Bausch and Lomb).

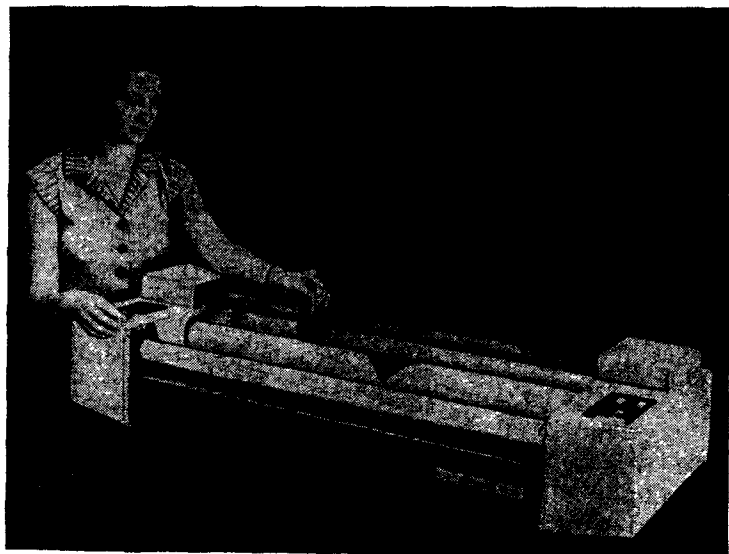


Рис. 23.6. Настольный трехперьевой рулонный графопостроитель (California Computer Products, Inc.).

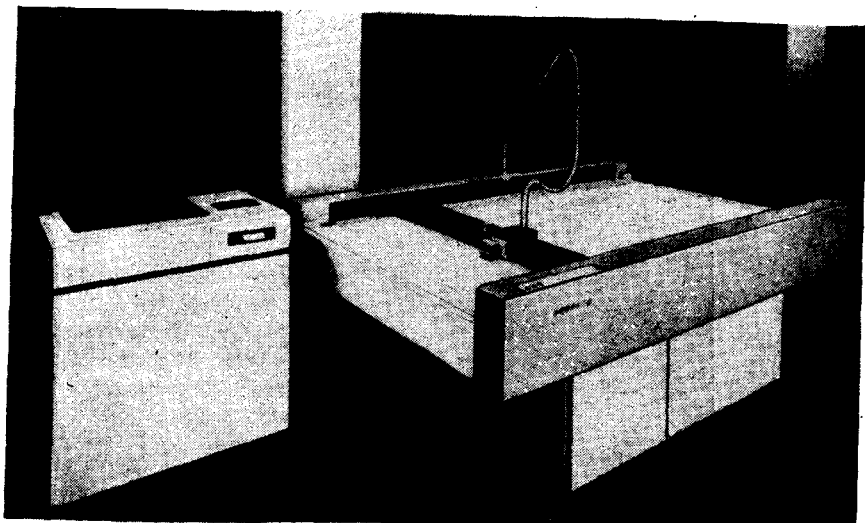


Рис. 23.7. Высокопроизводительный напольный графопостроитель для отрисовки графических изображений размером 1270×2083 мм (California Computer Products, Inc.).

бумаги. В **перьевых рулонных графопостроителях** перо движется в поперечном направлении, а бумага — в продольном направлении. Одновременные перемещения пера и бумаги позволяют изобразить любые желаемые линии, т. е. получить графическое изображение. Обычно бумага может двигаться в продольном направлении в обе стороны (вперед и назад), чем обеспечивается непрерывная отрисовка отдельного сегмента изображения; тем не менее во избежание излишних потерь времени на перемещения бумаги целесообразно графическое изображение, вытянутое в продольном направлении, сегментировать на участки небольшой длины.

В **растровых электростатических графопостроителях** движение бумаги происходит только вперед. Отрисовка изображения осуществляется в виде последовательных рядов близко расположенных друг к другу точек, причем все точки одного ряда выпечатываются одновременно.

Во всех графопостроителях, использующих движение бумаги, ширина рисунка ограничена или шириной барабана, или диапазоном перемещения пишущего узла (обычно от 279 до 914 мм), а длина практически бесконечна.

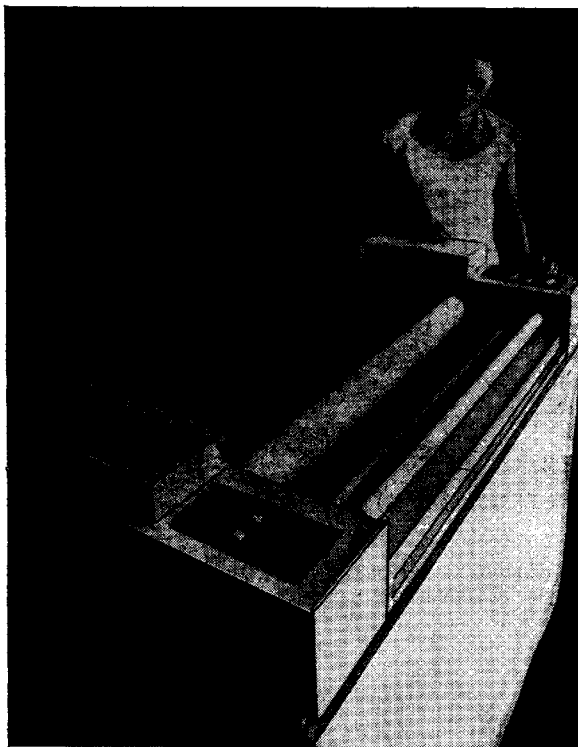


Рис. 23.8. Напольный четырехперьево́й руло́нный графопостроитель для отрисовки графических изображений шириной 864 мм и неограниченной длины (California Computer Products, Inc.).

23.4. КОНСТРУКЦИИ ГРАФОПОСТРОИТЕЛЕЙ

Размеры графического изображения для планшетных графопостроителей определяются максимальной высотой и шириной листа, а для рулонных устройств — только шириной. В дополнение к этому и тот и другой тип устройств может быть представлен двумя конструкциями — настольной и напольной. Настольная конструкция используется для отрисовки небольших чертежей или документов страничного формата. Напольная конструкция используется для отрисовки больших по размеру изображений: больших чертежей, плазов, карт и т. п. (рис. 23.7 и 23.8).

23.5. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ГРАФОПОСТРОИТЕЛЕЙ

Помимо уже перечисленных общих характеристик целесообразно рассмотреть следующие технические характеристики, определяющие физические возможности графопостроителей:

размер шага, длина и ширина чертежа, скорость работы — для шаговых перьевых графопостроителей;

плотность точек растра, скорость продвижения бумаги — для электростатических растровых графопостроителей.

Длина элементарного перемещения в шаговом перьевом графопостроителе может изменяться в высокоточных моделях от 2,540 до 0,003 мм. Зависимость точности воспроизведения линии от размера шага нагляднее всего проявляется для прямых, незначительно отличающихся от вертикали или горизонтали. На рис. 23.9 показано, как выглядит линия, имеющая небольшое отклонение от вертикали для трех значений шага. Хотя уменьшением шага можно достичь большей точности, при этом возрастают затраты времени на отрисовку, поскольку увеличивается число разгонов и торможений пишущего узла. Некоторую компенсацию таких затрат можно получить, увеличив скорость движения на каждом шаге.

Для настольного графопостроителя максимальный размер рисунка ограничен 178×254 мм, в то время как стандартный планшетный графопостроитель позволяет формировать изображения на листах размером $1,2 \times 1,5$ м. Специализированные планшетные графопостроители обеспечивают размеры изображения $1,5 \times 2,4$ м или $1,8 \times 7,3$ м. Они используются для отрисовки плазов и изготовления больших рабочих чертежей. Размеры изображения в рулонных графопостроителях ограничены только их шириной, составляющей обычно от 279 до 864 мм.

Скорости перемещения пера в перьевых графопостроителях изменяются от 240 до 2600 элементарных шагов в минуту (для самых быстрых устройств). Чаще всего эта скорость не превышает 1000 шагов в минуту.

У электростатических графопостроителей плотность точек лежит, как правило, в диапазоне от 4 до 8 точек на 1 мм,

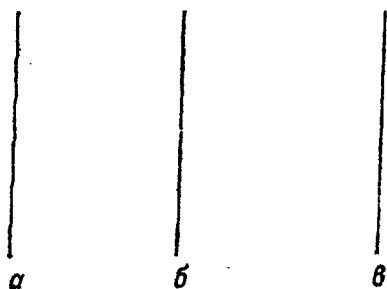


Рис. 23.9. Примеры отрисовки линии для трех значений шага: а) 0,25 мм, б) 0,125 мм, в) 0,032 мм. Прямые слегка наклонены к вертикали, что позволяет разглядеть неточности, вызванные значением шага (фирма Hewlett-Packard).



Рис. 23.10. Автоматизированная чертежная система для разработки печатных плат, включающая графопостроитель и видеомонитор (слева). (С разрешения фирмы Tektronix.)

а скорость движения бумаги составляет от 6,4 до 76,2 мм/с в зависимости от модели. Чаще всего последний показатель лежит в пределах от 25,4 до 50,8 мм/с.

23.6. ПРИМЕНЕНИЕ ГРАФОПОСТРОИТЕЛЕЙ

Графопостроители широко применяются для автоматического построения чертежей в составе автоматизированных чертежных систем. Пример такой системы, применяемой для разработки печатных плат, показан на рис. 23.10. Такие системы используются для изготовления чертежей конструкции, карт земной поверхности, электрических схем, печатных плат. Часто в составе чертежных систем используются кодировщики информации, позволяющие выполнять оцифровку чертежей, схем для их ввода в систему. Существуют, например, стереоскопические кодировщики, с помощью которых автоматизированная система может выдавать контурные карты непосредственно по стереофотографии земной поверхности с воздуха. Обычно в системах такого рода используются высокопроизводительные планшетные графопостроители с большой поверхностью поля. Необходимо различать графопостроители, используемые как специализиро-

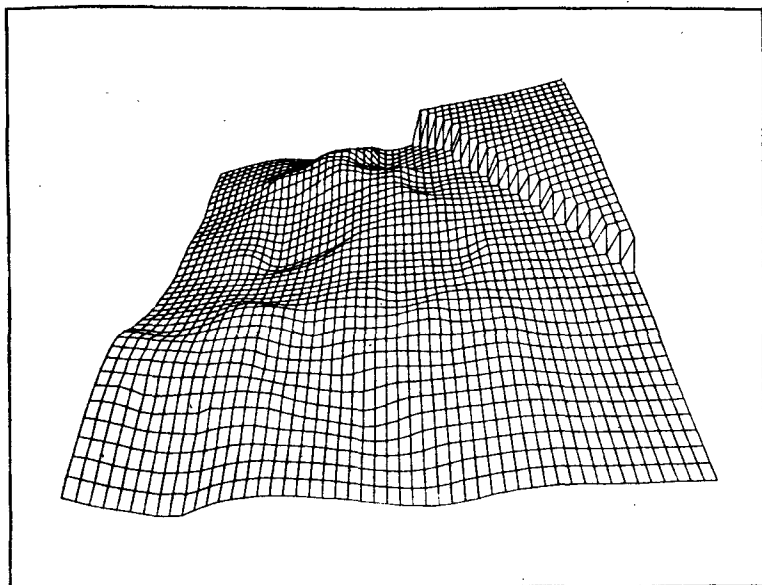


Рис. 23.11. Трехмерное изображение геологического разреза (сдвиг поверхности), выполненное графопостроителем по данным, полученным на ЭВМ (California Computer Products, Inc.).

ванные системы, от графопостроителей в качестве внешних устройств ЭВМ. Аналогично тому как некоторые виды внешних устройств могут играть главную роль в специализированных вычислительных системах, графопостроители в составе автоматизированных чертежных систем не являются лишь периферийным устройством главной ЭВМ, а служат центральным звеном автоматизированной системы.

23.7. ГРАФИЧЕСКИЕ ПРОГРАММЫ

Графопостроители как технические устройства вместе со специальными программами для ЭВМ позволяют получить разнообразные графические изображения, в том числе трехмерные изображения объектов (рис. 23.11), символьные изображения (рис. 23.12) и другие.

В простейшем случае инструкции для небольшого настольного графопостроителя планшетного типа позволяют осуществить перемещение пера в любом из восьми возможных направлений (рис. 23.13). Типичный набор таких инструкций в коде ASCII приведен на рис. 23.14. С помощью этих инструкций

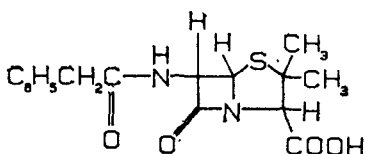


Рис. 23.12. Символьное изображение химической молекулы пенициллина, выполненное на графопостроителе (фирмы Hewlett-Packard).

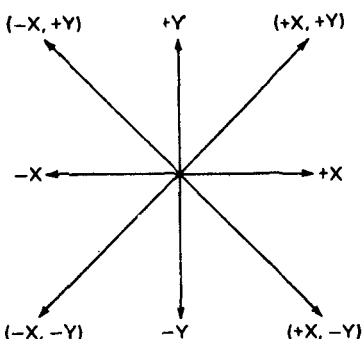


Рис. 23.13. Возможные направления перемещения пера, реализуемые в шаговом планшетном графопостроителе и их обозначения (отделение Houston Instrument фирмы Bausch and Lomb).

	Команды графопостроителя	Команды в коде ASCII
Направление движения пера	+Y +X, +Y +X +X, -Y -Y -X, -Y -X -X, +Y	p q r s t u v w
Команды вертикального пе- ремещения пера	ПОДНЯТЬ ПЕРО ОПУСТИТЬ ПЕРО	y z
Стабилизация пера	ЖДАТЬ	ПРОБЕЛ ()

Рис. 23.14. Инструкции для графопостроителя в коде ASCII, обеспечивающие перемещение пера в соответствии с рис. 23.13.

можно описать любое графическое изображение. Чтобы, например, нарисовать линию длиной 2,5 мм в положительном направлении оси X при длине элементарного шага, равной 0,25 мм, требуется выдать следующую последовательность инструкций: **ОПУСТИТЬ ПЕРО**; короткая пауза для стабилизации пера; десять команд $+X$. В приведенной кодировке последовательность будет выглядеть так:

z—ggggggggg.

Если микроЭВМ допускает программирование на языках Бейсик, Фортран или каком-нибудь другом языке высокого уровня, то могут быть написаны простые подпрограммы, автоматически рассчитывающие последовательность элементарных шагов, которую нужно выдать для построения линии, проходящей между любыми двумя точками на плоскости. Такие подпрограммы называются **генераторами векторов**; они избавляют программиста от необходимости писать точную последовательность инструкций, подобную приведенной выше.

Писать всякий раз набор подробных пошаговых инструкций графопостроителя для каждого коммерческого или научного приложения было бы непродуктивно; такая работа напоминала бы программирование на языке ассемблера. Поэтому обычно на языках высокого уровня пишутся специальные графические подпрограммы, позволяющие управлять работой графопостроителя, не прибегая к пошаговым командам; в набор этих подпрограмм входят простые генераторы векторов и процедуры для многократного повторения некоторых операций при рисовании. В этом разделе далее будут приведены типичные примеры таких графических программ.

Графические подпрограммы

Некоторые процедуры отрисовки одинаково необходимы для любых приложений, например отрисовка окружностей, прямоугольников или штриховых линий. Типовой набор графических подпрограмм общего назначения приведен в табл. 23.1.

Таблица 23.1. Набор графических подпрограмм общего назначения

Имя подпрограммы	Выполняемые функции
CIRCL	Отрисовка окружности, дуги или спирали
DASHL	Проведение штриховой линии
DASHP	Проведение штриховой линии до указанной точки
ELIPS	Отрисовка эллипса или дуги эллипса
FIT	Проведение кривой через три точки
GRID	Отрисовка прямоугольной сетки
POLY	Отрисовка правильного многоугольника
RECT	Отрисовка прямоугольника

Иллюстрация работы некоторых подпрограмм из этого списка приведена на рис. 23.15. Помимо генераторов векторов для отрисовки прямых, кривых и ряда фигур, имеются еще и генераторы букв — заглавных и строчных, — позволяющие выводить текст и аннотировать графики. В некоторых графопостроителях

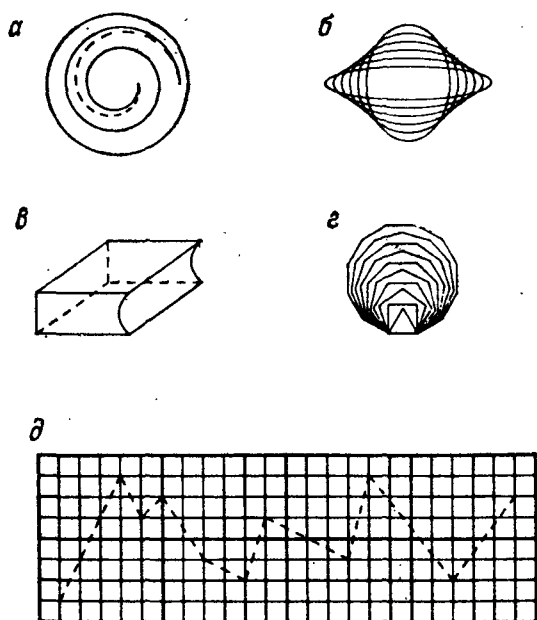


Рис. 23.15. Работа подпрограмм для отрисовки графических изображений (California Computer Products, Inc).

a — окружности спирали; *б* — эллипсы; *в* — штриховка невидимых линий; *г* — равносторонние многоугольники; *д* — прямоугольная сетка со штрихпунктирной линией.

5 Resident Character Set

ANSI ASCII

9825A ASCII

THREE EUROPEAN SETS

with programmable

SIZE *slant and*
direction

Рис. 23.16. Различные способы вывода текстовой информации на графопостроитель (фирма Hewlett-Packard).

есть возможность работать сразу с несколькими наборами символов, различающимися по размеру, наклону или направлению вывода текста (рис. 23.16).

Рис. 23.17 дает пример работы графической программы, которая вычерчивает оси, ставит под ними подписи, изображает два массива точек, соединяет их прямыми, штрихует область между двумя кривыми. В табл. 23.2 перечислены типовые подпрограммы, используемые для вычерчивания подобных графических изображений.

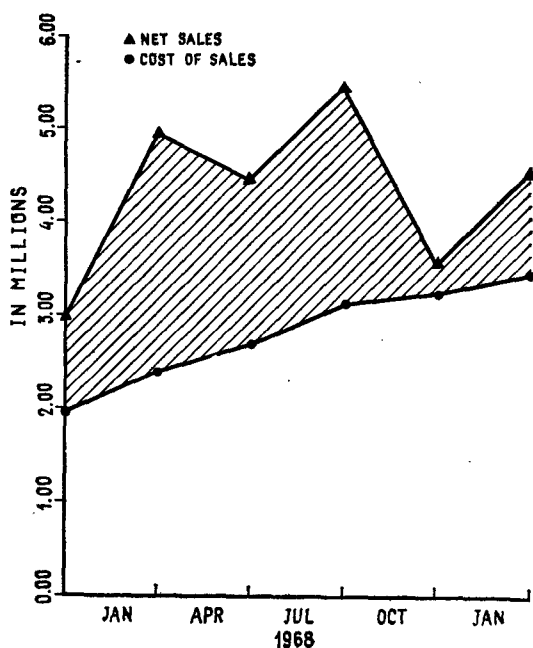


Рис. 23.17. Пример работы графической программы (California Computer Products, Inc.).

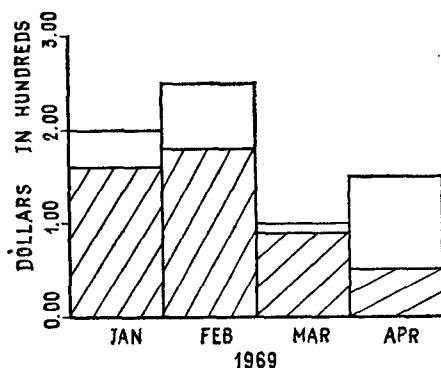


Рис. 23.18. Пример работы программы для отрисовки диаграмм (California Computer Products, Inc.).

ков, а на рис. 23.18 приведен пример использования подпрограммы BAR.

На рис. 23.19 приведен чертеж, составленный с помощью набора подпрограмм из табл. 23.3. При изображении различных схем, кроме разметочных функций, перечисленных в

Таблица 23.2. Набор графических подпрограмм для экономических приложений

Имя подпрограммы	Выполняемая функция
AXISB	Отрисовка оси ординат с разметкой
AXISC	Отрисовка оси абсцисс с календарной разметкой
BAR	Отрисовка диаграммы
HIST	Отрисовка гистограммы
LBAXS	Отрисовка логарифмической шкалы по оси ординат с разметкой
LGLIN	Отрисовка в логарифмическом или полулогарифмическом масштабах
PIE	Отрисовка круговой диаграммы
SHADE	Штриховка области между двумя линиями
SCALG	Масштабирование при переходе к логарифмическим координатам

Таблица 23.3. Набор графических подпрограмм для автоматизированных чертежных систем

Имя подпрограммы	Выполняемая функция
AROHD	Нанесение стрелки
ARROW	Нанесение линии со стрелкой на конце
CNTRL	Вычерчивание центральной линии
DIMEN	Вычерчивание размерной линии
LABEL	Нанесение надписи между указанными точками

табл. 23.3, часто требуется набор специальных символов или букв. Примерный перечень соответствующих графических подпрограмм приведен в табл. 23.4.

Таблица 23.4. Набор графических подпрограмм для изображения различных схем

Имя подпрограммы	Выполняемая функция
ELECT	Изображение электрического символа
GEO	Изображение геологического символа
GRAF	Отрисовка выбранного символа в заданной точке
GREEK	Изображение буквы греческого алфавита
MAP	Изображение картографического символа
MATH	Изображение математического символа
POWER	Изображение электротехнического символа
WELD	Изображение символа сварных соединений

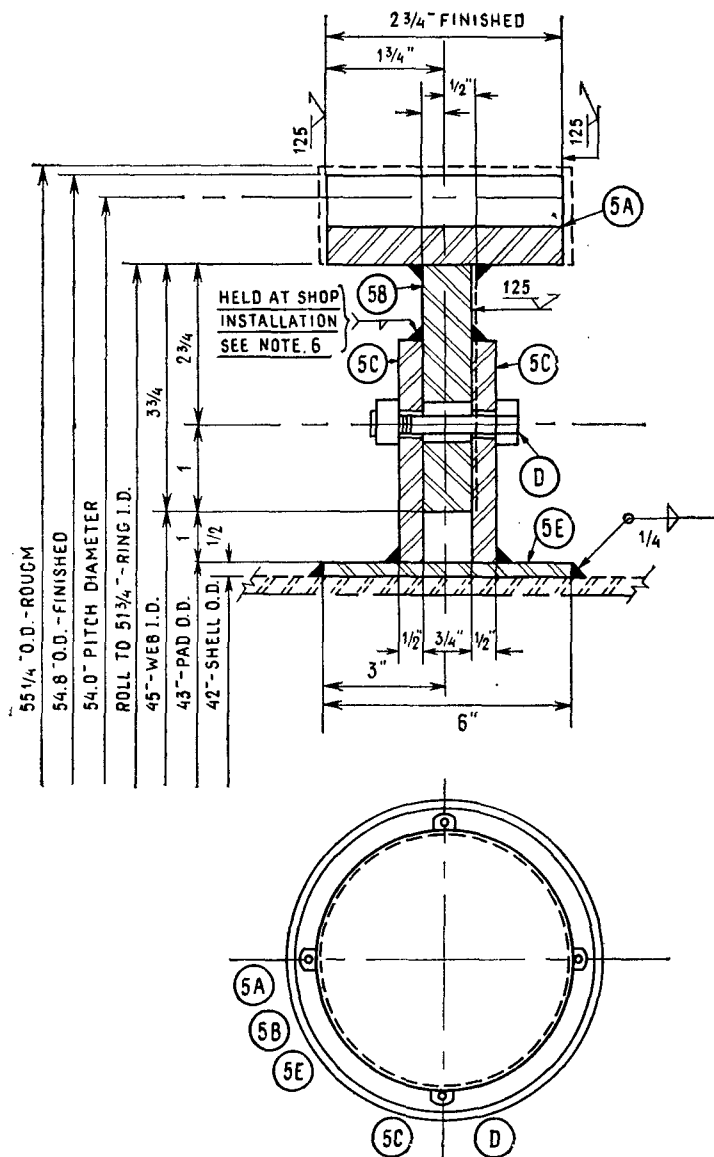


Рис. 23.19. Чертеж, выполненный с помощью набора графических подпрограмм (California Computer Products, Inc.).

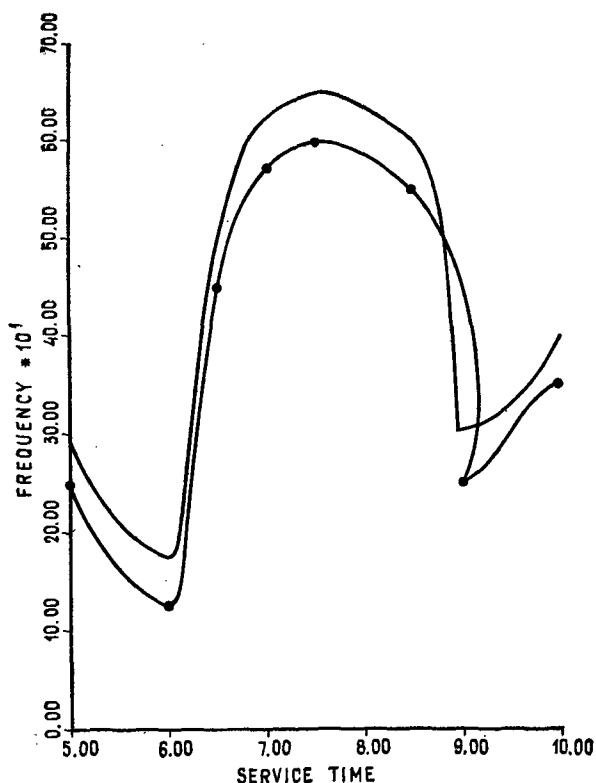


Рис. 23.20. Пример использования графических подпрограмм для интерполяции и аппроксимации по экспериментальным данным (California Computer Products, Inc.).

— аппроксимация, ● интерполяция.

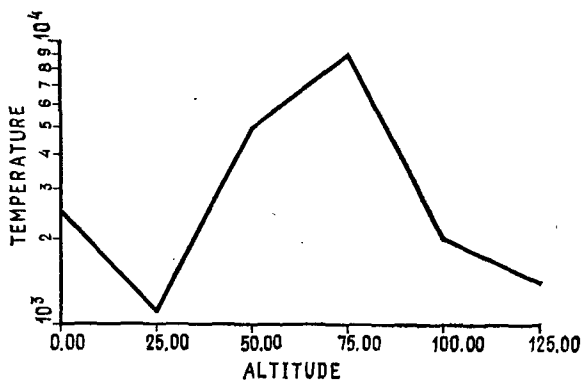


Рис. 23.21. Пример вычерчивания графика с логарифмической шкалой (California Computer Products, Inc.).

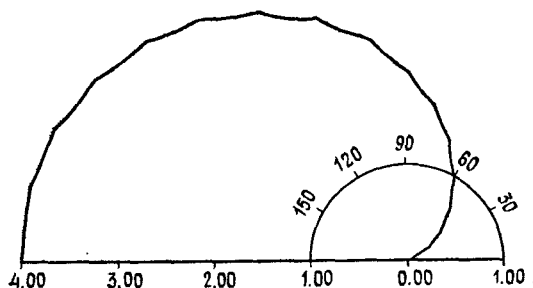


Рис. 23.22. Пример использования подпрограммы отрисовки в полярных координатах (California Computer Products, Inc.).

На рис. 23.20—23.22 показаны примеры использования графических подпрограмм для научных приложений, перечисленных в табл. 23.5.

Таблица 23.5. Набор графических подпрограмм для научных приложений

Имя подпрограммы	Выполняемая функция
CURVX	Отрисовка функции от аргумента X в заданном интервале
CURVY	Отрисовка функции от аргумента Y в заданном интервале
FLINE	Интерполяция на основе заданного набора точек
LGAXS	Отрисовка и разметка логарифмических осей
LGLIN	Нанесение данных в логарифмическом или полулогарифмическом масштабах
POLAR	Вывод точки в полярных координатах
SCALG	Масштабирование при переходе к логарифмическим координатам
SMOOT	Аппроксимация на основе заданного набора точек

На рис. 23.20 показано применение подпрограмм FLINE, SMOOT для интерполяции и аппроксимации по экспериментальным данным. На рис. 23.21 проиллюстрировано использование подпрограмм SCALG, LGAXS и LGLIN для вычерчивания графика в полулогарифмическом масштабе. На рис. 23.22 дан пример использования подпрограммы POLAR для построения кривых в полярных координатах.

Из сказанного ясно, что использование типовых графических подпрограмм позволяет получать с помощью графопостроителей твердые копии для разного рода прикладных задач, решаемых на мини- и микроЭВМ.

Мини- и микроЭВМ нужны еще и для того, чтобы преобразовать последовательность вызовов этих графических подпрограмм в инструкции графопостроителя. Если графопостроитель

относится к растровому типу, то ЭВМ должна, помимо прочего, для каждого графического изображения сформировать его точечный растр. Однако даже при увеличении объема вычислений производительность современных ЭВМ, как правило, все равно в 100 или 1000 раз превышает скорость работы графопостроителя.

Если миниЭВМ используется не только для графических работ, но и выполнения расчетов, то ее производительности может оказаться недостаточно для эффективной работы. В этом случае целесообразно графические инструкции для графопостроителя вывести на магнитную ленту. Затем лента переносится на автономный графический узел, который включает процессор, обеспечивающий считывание графических инструкций с магнитной ленты и управление графопостроителем. Такое решение избавляет от необходимости согласовывать скорость работы основной ЭВМ со скоростью графопостроителя.

*Д. Стоут***24.1. ЭВОЛЮЦИЯ МИКРОПРОЦЕССОРОВ**

Несколько областей электроники испытали период бурного развития только из-за того, что они были связаны с микропроцессорами, микроЭВМ и ассоциативными интегральными схемами. Сложность устройств на интегральных схемах удваивается примерно за год; эта тенденция прослеживается с начала 60-х годов — именно в это время появились первые устройства на интегральных схемах. Если такая тенденция сохранится, то устройства, содержащие сотни миллионов транзисторов в одном кристалле, появятся уже в 90-е годы. На рис. 24.1 приведен график, характеризующий рост плотности размещения элементов в кристалле за последние двадцать лет. Такой же стремительный рост уровня интеграции ожидается и в ближайшее десятилетие, если удастся преодолеть существующие на сегодняшний день технологические трудности. Многие препятствия, ранее стоявшие на этом пути, были устранены благодаря появлению современной достаточно совершенной технологии производства интегральных схем. Можно с уверенностью сказать, что и в дальнейшем электроника интегральных схем будет быстро развиваться, одна технология будет сменять другую, как только возможности последней будут исчерпаны.

В начале 60-х годов впервые был изготовлен логический элемент на кремниевом кристалле. Как только появилась возможность разместить несколько элементов в кристалле, усложнились и функции устройств, построенных на этих кристаллах. Уже через несколько лет после того, как был создан первый кристалл, пользователь с помощью набора интегральных схем (ИС) мог реализовать почти все логические и арифметические функции, выполняемые современными микропроцессорами. С помощью кристалла арифметико-логического устройства (АЛУ), разработанного в 70-е годы, можно было осуществлять

¹⁾ Adapted from Microprocessor Applications Handbook, edited by David F. Stout. Copyright © 1982. Used by permission of McGraw-Hill, Inc. All rights reserved.

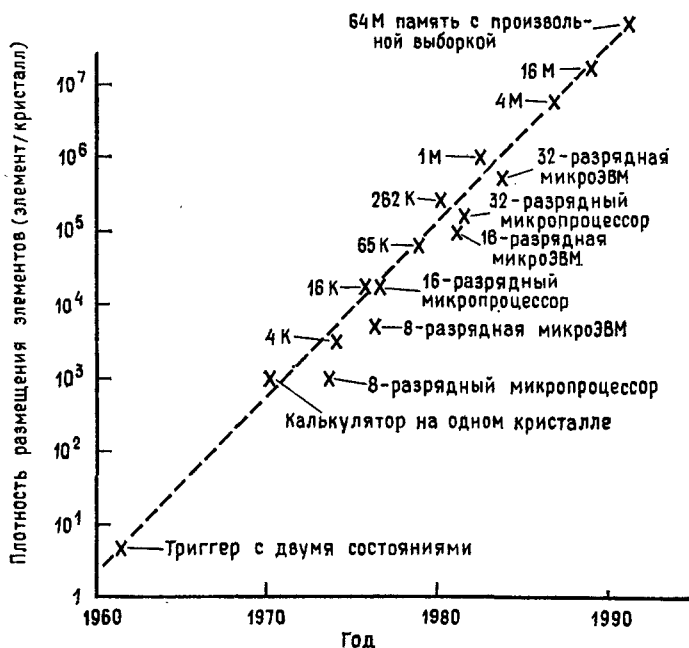


Рис. 24.1. Изменение плотности размещения элементов в кристалле за период 1960—90 гг. (взято из работы [1] с разрешения).

операции сложения, вычитания, простого и циклического сдвига, т. е. как раз те операции, которые выполняют современные микропроцессоры и микроЭВМ. Однако только в начале 70-х годов появилась возможность разместить в одном кристалле АЛУ и элементы последовательного действия (триггеры, регистры сдвига и т. п.); так был создан микропроцессор. Это стало возможным, когда степень интеграции достигла 1000 транзисторов на кристалл. Не прошло и нескольких лет, а в одном кристалле удалось разместить схемы ввода-вывода, постоянное запоминающее устройство (ПЗУ), память с произвольной выборкой и схемы синхронизации. Сформированное таким образом устройство получило название микроЭВМ. Граница, разделяющая понятия микропроцессор и микроЭВМ, точно не определена. Считается, что микропроцессор только выполняет команды, а микроЭВМ не только выполняет команды, но и

- организует связь с устройствами ввода-вывода, работающими как в параллельном, так и в последовательном режимах;
- имеет обычное или программируемое ПЗУ, в котором хранятся программы пользователей;

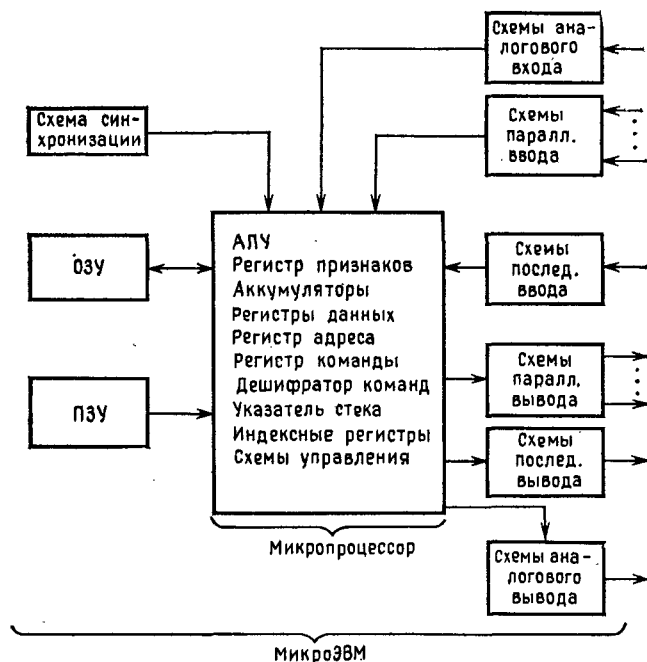


Рис. 24.2. Упрощенная схема микроЭВМ.

— имеет оперативное запоминающее устройство (ОЗУ) большого объема, предназначенное для временного хранения данных и программ;

— обеспечивает с помощью схем синхронизации согласованную работу всех входящих в нее устройств.

МикроЭВМ иногда называют «ЭВМ в кристалле». Следует отметить, что этот термин на самом деле не очень удачен, поскольку независимо от того, сколько логических элементов удастся разместить в кристалле, потребуется еще клавиатура, дисплей и источник питания. Для создания многих систем на базе микроЭВМ будут нужны еще и специальные кристаллы для организации связи с внешней средой. В этом случае главным препятствием является ограниченное число выводов у микросхемы, в корпусе которой размещена микроЭВМ. Хотя кристаллы, содержащие миллионы элементов, будут созданы уже в 80-е годы, главной задачей, возникающей при их использовании, будет организация связи с внешней средой.

Анализ рис. 24.2 показывает, что микропроцессор является основным компонентом микроЭВМ. Одни микроЭВМ включают больше компонентов, чем представлено на рис. 24.2, другие

меньше. Многие специализированные микроЭВМ используют выполненные в одном кристалле специальные интерфейсы ввода-вывода для подключения нестандартных внешних устройств.

24.2. КОМПОНЕНТЫ МИКРОПРОЦЕССОРА

Арифметико-логическое устройство. Как уже отмечалось, родоначальником микропроцессоров является АЛУ. Например, АЛУ 74181 — четырехразрядное устройство, состоящее из 75 элементов. Оно может выполнять 16 базовых операций над двумя четырехбитовыми входными словами. Результат помещается в четырехбитовое выходное слово и подается на четыре дополнительные выходные линии. Функция SELECT оперирует с четырьмя входными линиями. На рис. 24.3 представлена схема АЛУ в виде черного ящика. Объединяя несколько АЛУ, можно работать с входными и выходными словами любого размера.

Таблица 24.1. Набор операций АЛУ 74181¹⁾

Функция SELECT				Арифметическая функция $M=C_n=0$	Логическая функция $M=1$
S3	S2	S1	S0		
0	0	0	0	$F = A$	$F = \bar{A}$
0	0	0	1	$F = A + B$	$F = \bar{A} + \bar{B}$
0	0	1	0	$F = A + \bar{B}$	$F = \bar{A} \cdot B$
0	0	1	1	$F = -1$ (дополнение до 2)	$F = 0000$
0	1	0	0	$F = A$ плюс $A \cdot \bar{B}$	$F = \bar{A} \cdot \bar{B}$
0	1	0	1	$F = (A + B)$ плюс $A \cdot \bar{B}$	$F = \bar{B}$
0	1	1	0	$F = A - B - 1$	$F = A \oplus B$
0	1	1	1	$F = A \cdot \bar{B} - 1$	$F = A \cdot \bar{B}$
1	0	0	0	$F = A$ плюс $A \cdot B$	$F = \bar{A} + B$
1	0	0	1	$F = A$ плюс B	$F = \bar{A} \oplus B$
1	0	1	0	$F = (A + \bar{B})$ плюс $A \cdot B$	$F = B$
1	0	1	1	$F = A \cdot B - 1$	$F = A \cdot B$
1	1	0	0	$F = A$ плюс $A = 2A$	$F = 1111$
1	1	0	1	$F = (A + B)$ плюс A	$F = A + \bar{B}$
1	1	1	0	$F = (A + \bar{B})$ плюс A	$F = A + B$
1	1	1	1	$F = A - 1$	$F = A$

¹⁾ Здесь знак «+» соответствует логическому ИЛИ, «-» логическому И, « \oplus » логическому ИСКЛЮЧАЮЩЕЕ ИЛИ (взято из работы [1]).

В табл. 24.1 представлены 16 функций, реализуемых АЛУ 74181. Если значение сигнала на управляющем входе M равно 0, то выполняются арифметические операции; если же $M = 1$, то выходное слово F представляет результат одной из 16 логических операций над словами A и B . В том случае, когда

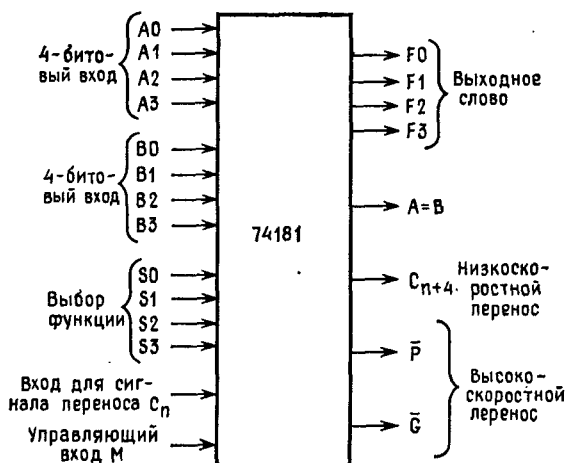


Рис. 24.3. Логическая схема АЛУ 74 181 (взято из работы [1] с разрешения).

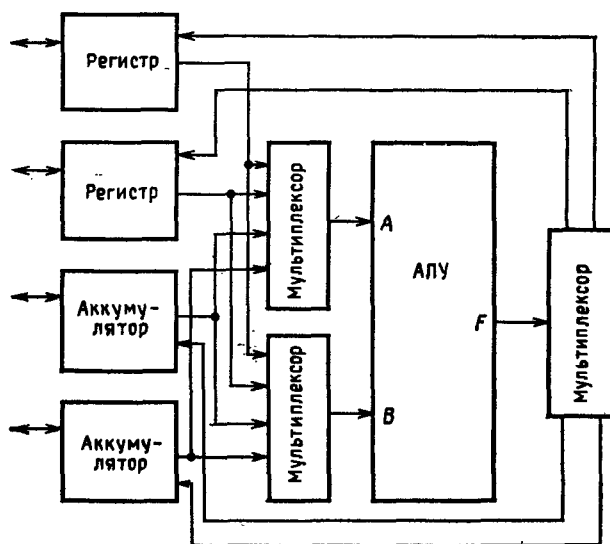


Рис. 24.4. Схема микропроцессора, включающая АЛУ, регистры, аккумуляторы и мультиплексоры передачи данных. Каждая линия на схеме обозначает шину данных, состоящую из 4, 8 или 16 проводников (взято из работы [1] с разрешения).

используется устройство, выполняющее как операцию сложения, так и логическую операцию ИЛИ, необходимо проявлять осторожность при употреблении символа (+). В дальнейшем для обозначения операции ИЛИ будет использоваться знак \vee , а для обозначения операции сложения — слово плюс. В табл. 24.2 приведены базовые операции АЛУ общего назначения.

Таблица 24.2. Основные операции АЛУ¹⁾ общего назначения

Функция	Описание функции (в любой операции А и В можно поменять местами)
$F = A \text{ плюс } 1$	Увеличение А (или В) на 1
$F = A - 1$	Уменьшение А (или В) на 1
$F = A \text{ плюс } B$	Сложение А и В
$F = A - B$	Вычитание В из А
$F = A \cdot B$	Вычисление логической функции И
$F = A + B$	Вычисление логической функции ИЛИ
$F = A \oplus B$	Вычисление функции ИСКЛЮЧАЮЩЕЕ ИЛИ
$F = 2A$	Сдвиг А влево
$F = A/2$	Сдвиг А вправо
$F = \text{цикл. сдвиг влево}$	Циклический сдвиг А влево
$F = \text{цикл. сдвиг вправо}$	Циклический сдвиг А вправо
$F = A$	Дополнение А
$F = 0 \cdot A$	Очистка А

¹⁾ Считается, что имеются две входные шины А и В и одна выходная шина F.

Регистры. АЛУ микропроцессоров обычно обладают несколько большими возможностями, чем автономные АЛУ. Дополнительные возможности связаны с подключением к АЛУ нескольких регистров памяти (рис. 24.4). Одновременно АЛУ может взаимодействовать только с двумя регистрами. Следует отметить, что в микропроцессорах предусмотрено несколько специальных регистров, называемых аккумуляторами, или накапливающими регистрами. Они хранят результаты предшествующих операций для их последующего использования.

Мультиплексоры. Эти устройства предназначены для передачи данных из выбранного регистра во входной порт АЛУ. Еще один мультиплексор передает выходные данные АЛУ в регистр, где они хранятся до тех пор, пока не будут затребованы каким-либо внешним устройством. Управление выполнением этих операций осуществляется с помощью последовательной схемы, которая связана со всеми регистрами и мультиплексорами. В качестве примера укажем, что в 8-разрядном микропроцессоре регистры, аккумуляторы и мультиплексоры являют-

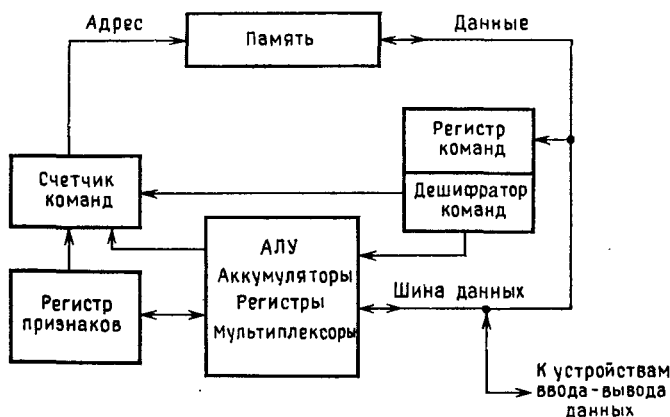


Рис. 24.5. Структура микропроцессора с полным набором функций.

ся 8-разрядными устройствами с параллельным входом и выходом.

Следующим шагом в создании микропроцессора с полным набором функций было дополнение его счетчиком команд, регистром признаков, регистром команд, дешифратором команд и памятью. Поскольку все эти устройства взаимосвязаны, они могут быть добавлены лишь одновременно. Связи, существующие между этими устройствами, показаны на рис. 24.5. Если речь идет о микропроцессоре, то память, как правило, отделена от него. Если же рассматривается микроЭВМ, то большая часть памяти обычно бывает расположена на одном кристалле с микропроцессором.

Счетчик команд. Стандартные 8-разрядные микропроцессоры имеют 16-разрядный счетчик команд. В счетчике команд хранится число, являющееся порядковым номером выполняемой в данный момент команды программы. Точнее говоря, хранится не номер, а адрес участка памяти, где эта команда расположена. 16-разрядный счетчик команд позволяет выполнять программы, длина которых не превышает 65 535 команд. Реальные программы имеют длину в несколько сотен или тысяч команд, так что предел, равный 65 535, достигается крайне редко.

Память и регистр команд. Как видно из рисунка 24.5, программный счетчик непосредственно не связан с работой АЛУ. Содержимое программного счетчика позволяет лишь определить место хранения нужного оператора программы в оперативной памяти и передать его в виде слова в регистр команд. Там это слово неходится до тех пор, пока дешифратор команды не распознает, какую операцию должна выполнить АЛУ и(или) счет-

чик команд. Если дешифратор команд распознает команду ветвления или перехода, то счетчик команд получит новое значение, которое и будет определять адрес следующей выполняемой команды. В этом случае АЛУ не выполняет никаких действий. Однако если дешифратор команд распознает команду из числа приведенных в табл. 24.1 или 24.2, то АЛУ и размещенные в одном блоке с ним регистры начинают работу. Данные передаются в АЛУ и из него по шине данных, состоящей из 4, 8 или 16 параллельных линий. Эту же шину данных использует и оперативная память. Данные могут передаваться в любом направлении по этой шине. Работой шины управляет последовательная логическая схема. В большинстве систем шина данных может находиться в трех состояниях. Все источники и приемники, связанные с шиной, обычно находятся в пассивном состоянии. В процессе выполнения очередной команды дешифратор команд распознает, какие источники и приемники информации перевести в активное состояние. Во время выполнения команды по шине передается информация только между этими двумя устройствами.

Регистр признаков (РП). Во время работы АЛУ используется специальный регистр, называемый регистром признаков (РП). Значение каждого бита этого регистра зависит от результатов операций, выполненных АЛУ к текущему моменту. Например, если во всех разрядах регистра АЛУ находятся нули, бит РП, называемый нулевым битом (Z-бит), устанавливается в 1. Если возникает перенос, то бит С устанавливается в 1, если нет — то бит NC устанавливается в 1. Этот регистр называют также либо регистром состояния, либо регистром флажков.

Существует большое число различных типов флажков: в табл. 24.3 приведены наиболее распространенные. Поскольку большинство 8-разрядных микропроцессоров используют 8-разрядный регистр признаков, список флажков, приведенный в табл. 24.3, шире обычно используемого. В некоторых микропроцессорах используется лишь шесть флажков.

Выполнение команд перехода зависит от состояния регистра признаков. Например, если выполняется команда ПЕРЕХОДА ПО УСЛОВИЮ НЕ РАВНО, дешифратор команд должен дать возможность АЛУ завершить выполняемую операцию перед тем, как новые команды будут переданы с помощью счетчика команд. Если результат не равен нулю, то дешифратор команд сформирует новый начальный адрес для счетчика команд. Если результат равен нулю, счетчик команд увеличивает свое значение на единицу. Та часть логического механизма, которая декодирует содержимое РП и рассылает управляющие сигналы различным устройствам микропроцессора, называется **блоком**

Таблица 24.3. Описание битов (флажков) регистра признаков

Имя бита (флажка)	Описание результата выполнения операции
Z	Результат $= 0$
NZ	Результат $\neq 0$
C	Наличие в результате единицы переноса
HC	Наличие в результате единицы переноса (для операций в двоично-десятичном коде)
NC	Отсутствие единицы переноса
B	Наличие отрицательного переноса
NB	Отсутствие отрицательного переноса
E	Равенство двух входных слов
P	Результат больше нуля
N	Результат меньше нуля
PA	Результат содержит нечетное число 1
OV	Переполнение регистра
I	Возникновение прерывания
GT	Значение на первом входе больше значения на втором входе
GE	Значение на первом входе больше или равно значению на втором входе
LT	Значение на первом входе меньше значения на втором входе
LE	Значение на первом входе меньше или равно значению на втором входе

Взято из работы [1].

условных переходов. Это один из наиболее сложных логических блоков микропроцессора.

Устройство управления. Логические схемы, выполняющие основные функции по управлению работой микропроцессора, обычно объединяются в устройство управления. Оно включает счетчик команд, регистр команд и дешифратор команд. Во время выполнения каждой команды устройство управления организует связи между регистрами.

Для реализации каждой команды может потребоваться до 10 последовательных шагов. Эти внутренние шаги называются микроциклами; каждый шаг соответствует обычно периоду генератора тактовых импульсов.

Во время каждого микроцикла выполняется одна микрокоманда. Промежуточные результаты могут сохраняться в регистрах в течение нескольких микроциклов. Иногда для этого выделяется специальный регистр, располагаемый между счетчиком команд и памятью. Этот регистр называется **регистром адреса (РА)**. Кроме того, между памятью и регистром команд часто размещается регистр, называемый **регистром данных (РД)**. Обычно для каждого микропроцессора разрабатывается много различных команд, и устройство управления должно различать, какой набор микрокоманд следует выполнять для ре-

ализации той или иной команды. Микрокоманды хранятся в управляющем ПЗУ или микроПЗУ. У большинства микропроцессоров ПЗУ расположено в том же кристалле, что и сам микропроцессор, но некоторые микропроцессоры имеют отдельный кристалл ПЗУ. Это позволяет проектировщикам в случае необходимости изменять некоторые микрокоманды. Микропроцессоры такого типа называются «микропрограммируемыми». Поскольку в микроПЗУ используется последовательная адресация, требуется счетчик команд, называемый **микропрограммным счетчиком**, который имеет разрядность не более трех-четырех битов.

Индексный регистр. С целью повышения эффективности программирования в большинство современных микропроцессоров включают один или несколько индексных регистров. В 8-разрядных микропроцессорах обычно используются 16-разрядные индексные регистры. Успешно использовать индексные регистры может только хороший программист. Их применение позволяет автоматически просматривать участки памяти (например, анализировать записи, находящиеся в таблице данных), в то время как счетчик команд занят выполнением команд основной программы. Таким образом, получается, что одновременно работают как бы два счетчика команд. С помощью 16-разрядного индексного регистра можно обеспечить адресацию памяти объемом 65 535 слов.

Указатель стека (УС). Указатель стека — это регистр, функции которого отличаются от функций всех остальных регистров. Возможности, предоставляемые этим регистром, могут быть полностью использованы только программистами высокой квалификации. В большинстве 8-разрядных микропроцессоров указатель стека содержит 16 бит. Поэтому он позволяет обратиться к участку памяти любого размера. Стек, адресуемый с помощью УС, — это произвольная группа последовательных элементов оперативной памяти. Эта память может и не располагаться в том же кристалле, что и сама микроЭВМ. Выделение определенной области оперативной памяти под стек планируется программистом, и это должно быть сделано в начальной части программы, где выделяется память под переменные. С помощью стека легко осуществлять хранение и поиск последовательно расположенных данных. При размещении блоков данных длиной в несколько слов желательно, чтобы микропроцессор получал команду прерывания от внешнего устройства. Для этого необходима специальная линия связи с внешним устройством, называемая **линией прерывания**. В том случае, когда периферийное устройство нуждается в обслуживании со стороны микропроцессора, оно посылает сигнал по линии прерывания, заставляя микропроцессор приостановить текущую

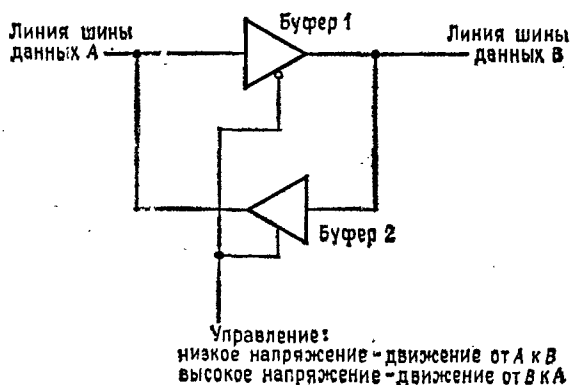


Рис. 24.6. Двунаправленный порт, позволяющий передавать данные в двух направлениях. Такие порты используются в микропроцессорах для организации интерфейса между внутренней шиной и внешними устройствами (взято из работы [1] с разрешения).

работу. Отметим, что счетчик команд хранит адрес стека, в котором находится информация, достаточная для возобновления выполнения прерванной программы после окончания обслуживания периферийного устройства. После получения сигнала прерывания микропроцессор сохраняет состояния регистра признаков, всех аккумуляторов, всех индексных регистров и т. д. Когда выполнение программы обслуживания периферийного устройства завершается, во все выше перечисленные регистры заносятся прежние значения и запускается прерванная программа.

Стек также используется для запоминания старого адреса программного счетчика, когда программа обращается к подпрограмме. В конце подпрограммы команда возврата заносит в программный счетчик число, на единицу большее хранимого в стеке.

Ввод-вывод. В микропроцессорах для передачи данных между регистрами используется внутренняя шина данных. Для обмена информацией между этой шиной и внешними устройствами нужны буферы, способные передавать информацию в обоих направлениях¹⁾. Часто и внутренние регистры снабжены такими портами для обмена информацией с устройствами, подключенными к внутренней шине. На рис. 24.6 приведена схема двунаправленного порта. Такие же порты используются и вне микропроцессора для передачи данных к внешним

¹⁾ Такие буферы в дальнейшем именуются двунаправленными портами. — *Прим. ред.*

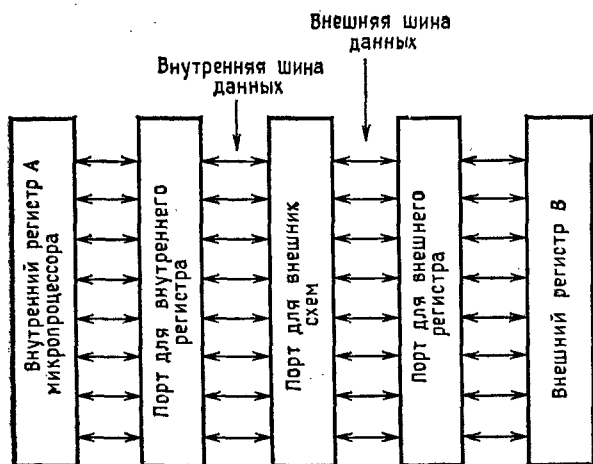


Рис. 24.7. Применение нескольких двунаправленных портов для передачи данных между регистрами микропроцессора и регистрами внешних устройств. В некоторых приложениях могут применяться однонаправленные порты (взято из работы [1] с разрешения).

устройствам. Поэтому для передачи данных от внутреннего регистра микропроцессора к регистру внешнего устройства может потребоваться до трех двунаправленных портов (рис. 24.7). Специальные программные средства осуществляют контроль за направлением передачи данных. С помощью рис. 24.7 можно проследить выполнение команды пересылки данных из регистра А в регистр В (STORE A, B). В процессе ее выполнения все три порта одновременно передают информацию слева направо, в результате через несколько микросекунд получаем $B = A$.

Внешняя память. При использовании микропроцессора необходимы внешние устройства для хранения констант, данных и команд. В свою очередь в однокристалльной микроЭВМ память для данных и команд размещается на кристалле, а константы и микропрограммы хранятся на внешних ПЗУ, возможно перепрограммируемых. Тем не менее программирование как для микропроцессоров, так и для микроЭВМ осуществляется одинаково. Однако спроектировать внешнюю память значительно труднее, чем пользоваться памятью, имеющейся в однокристалльной микроЭВМ.

При проектировании интерфейса микропроцессора с внешней памятью необходимо учитывать множество факторов. В первую очередь необходимо решить вопрос о размерах пространства внешней памяти. Затем надо разработать схему ад-

ресации, которая будет синхронизировать работу адресной шины с шиной данных, линией записи-считывания и некоторыми другими сигналами.

24.3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МИКРОПРОЦЕССОРОВ И МИКРОЭВМ

Каждый микропроцессор или микроЭВМ рассчитан на понимание одного языка. Причем для разных типов микропроцессоров эти языки несколько отличаются. Все они являются расширениями списка основных команд ЭВМ, приведенных в табл. 24.2. С точки зрения программиста команды микропроцессора могут быть разделены на три класса:

1. **Пересылка данных.** Ввод-вывод, Загрузка, Запись, Обмен, Передача.

2. **Обработка данных.** Сложение, Вычитание, Умножение, Деление, Дополнение, Очистка, Сдвиг, Циклический сдвиг, Приращение, Уменьшение на 1, И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ.

3. **Управление.** Безусловный переход, Пропуск, Условный переход, Вызов, Возврат, Задержка.

Рассмотрим более подробно каждый класс.

Пересылка данных. Данные можно передавать различными способами как внутри, так и вне микропроцессора. На рис. 24.8 показаны наиболее распространенные способы пересылки данных. Некоторые из них требуют для реализации нескольких команд, но в большинстве микропроцессоров основная часть операций по пересылке данных осуществляется одной командой. Данные могут пересылаться во внутренние регистры, оперативную или внешнюю память. Считается, что специализированные внешние устройства, такие, как порты ввода-вывода, таймеры и т. п., относятся к внешней памяти. Можно выделить четыре способа (типа) пересылки данных:

Первый тип: регистр — регистр.

Второй тип: регистр — память (включая выходные порты).

Третий тип: память — регистр (включая входные порты).

Четвертый тип: память — память.

Пересылки первых трех типов производятся одной командой микропроцессора. В первом и втором поколениях микропроцессоров для выполнения пересылки типа память — память требовалось две команды: одна — третьего типа, вторая — второго типа. Микропроцессоры третьего и четвертого поколений могут выполнять пересылку типа память — память одной командой.

Для различных микропроцессоров реализовано большое разнообразие команд пересылки, основанных на вышеупомяну-

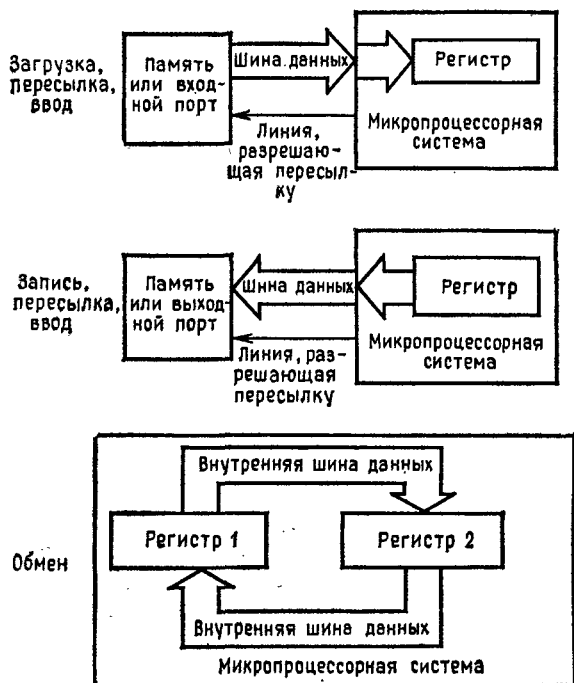


Рис. 24.8. Схемы пересылки данных в микропроцессорных системах (взято из работы [1] с разрешения).

тых базовых способах. Например, микропроцессор третьего поколения Z80 имеет семь команд для выполнения пересылок типа регистр — регистр, девять команд для пересылок типа регистр — память, семнадцать команд для пересылок типа память — регистр, и пять команд для пересылок типа память — память. В нем также предусмотрено десять команд обмена, каждая из которых является командой типа регистр — регистр, обеспечивающих передачу данных в обоих направлениях. Но и это еще не все. Некоторые команды, такие, как LDr, г', предусматривают 64 возможные комбинации г и г'. Приведенная выше команда загружает в любой из 8 г-регистров содержимое любого из 8 г'-регистров. Если учесть все возможные комбинации, то можно считать, что в микропроцессоре Z80 имеется 188 команд для пересылки данных. Микропроцессоры четвертого поколения, такие, как 16-разрядный микропроцессор 68 000, снабжены еще более гибким набором команд.

Способы адресации. Перед обсуждением двух других классов команд пересылки необходимо остановиться на способах адресации. Большинство 8-разрядных микропроцессоров и микроЭВМ используют 16-битовое адресное слово, позволяющее обращаться к $2^{16} = 65\,536$ позициям памяти. Если команда реализует обращение к памяти, используя способ расширенной адресации, то все 16 бит адреса должны непосредственно примыкать к 8-битовому коду команды. Такая 3-байтовая команда определяет наименее эффективный способ адресации. В случае прямой адресации адрес непосредственно следует за кодом команды. Прямая адресация может быть использована только тогда, когда искомый адрес принадлежит одному из 256 первых слов памяти. Для этого требуется расположить 8-битовое слово непосредственно за кодом команды. Это 8-битовое слово служит для выбора одного из 256 слов памяти, принадлежащих первой странице (т. е. с его помощью возможна адресация к первым 256 словам памяти).

Способ адресации с индексированием позволяет сформировать 16-битовый адрес, не используя следующие за командой биты. Для этого применяется 16-разрядный индексный регистр, содержащий нужный адрес памяти. С помощью специальных команд в этот регистр можно загружать данные, уменьшать или увеличивать его содержимое на 1, проверять, не равно ли оно нулю. Рассмотрим пример, когда необходимо загрузить в аккумулятор А числа, расположенные в памяти, начиная с адреса 4100 и кончая адресом 43FF. Для загрузки числа в аккумулятор А используется специальная подпрограмма. Если для этой цели применить микропроцессор 6800, то программа будет иметь следующий вид (отметим, что за символом # следует конкретное данное, а не адрес, а символ \$ используется для обозначения шестнадцатеричных данных):

LDX	#\$4100	;ЗАГРУЗИТЬ В ИНДЕКСНЫЙ
		;РЕГИСТР АДРЕС, РАВНЫЙ 4100
LOOP LDA	A, X	;ЗАГРУЗИТЬ В АККУМУЛЯТОР
		;А ДАННЫЕ, ВЗЯТЫЕ ИЗ
		;ОБЛАСТИ ПАМЯТИ, АДРЕС
		;КОТОРОЙ ХРАНИТСЯ В
		;ИНДЕКСНОМ РЕГИСТРЕ X
JSR	NAME	;ПЕРЕЙТИ К ПОДПРОГРАММЕ,
		;ГДЕ ИСПОЛЬЗУЕТСЯ ЗНАЧЕ-
		;НИЕ АККУМУЛЯТОРА А
INX		;УВЕЛИЧИТЬ НА 1 ЗНАЧЕНИИ
		;ИНДЕКСНОГО РЕГИСТРА

```

CPX    #$43FF ;СРАВНИТЬ ЗНАЧЕНИЕ X
        ;C 43FF
BNE    LOOP   ;ПЕРЕЙТИ НА МЕТКУ LOOP,
        ;ЕСЛИ X  $\neq$  43FF

```

Хотя в микропроцессоре 8080 фирмы Intel нет индексного регистра, способ косвенной адресации позволяет программисту выполнять адресацию с индексированием. Например, предыдущая программа для микропроцессора 8080 фирмы Intel будет выглядеть следующим образом:

```

LXIB    #$4100 ;ЗАГРУЗИТЬ В ПАРУ РЕГИ-
        ;СТРОВ ВС АДРЕС 4100
LOOP LDAX B    ;ЗАГРУЗИТЬ АККУМУЛЯТОР A
        ;ДАННЫМИ, АДРЕС КОТОРЫХ
        ;ХРАНИТСЯ В ПАРЕ РЕГИСТРОВ
        ;ВС
CALL    NAME   ;ПЕРЕЙТИ К ПОДПРОГРАММЕ,
        ;ГДЕ ИСПОЛЬЗУЕТСЯ ЗНАЧЕ-
        ;НИЕ АККУМУЛЯТОРА A
INXB    ;УВЕЛИЧИТЬ НА 1 ЗНАЧЕНИЕ,
        ;ХРАНЯЩЕЕСЯ В ПАРЕ РЕГИ-
        ;СТРОВ ВС
LXIN    #$-43FF ;ЗАГРУЗИТЬ В ПАРУ РЕГИСТ-
        ;РОВ HL ЗНАЧЕНИЕ - 43FF
DADB    ;HL = HL + BC
JNC     LOOP   ;ПЕРЕЙТИ НА МЕТКУ LOOP,
        ;ЕСЛИ BC  $\neq$  - HL

```

В этом примере пара регистров **BC** используется как 16-разрядный индексный регистр. В микропроцессоре 8080 фирмы Intel не предусмотрено 16-разрядных операций сравнения и вычитания, но есть команда 16-разрядного сложения, носящая название **DAD**. Эта команда осуществляет сложение содержимого пары регистров (в данном случае регистров **B** и **C**) с содержимым регистров **H** и **L**. Начальное значение, хранящееся в паре регистров **HL**, равно $-43FF$. Если $BC = 43FF$, то вырабатывается единица переноса и выполняется команда перехода.

Обработка данных. Реальная мощность микропроцессора определяется наличием большого числа разнообразных команд обработки данных. Многие проектировщики программного обеспечения и аппаратуры отдают предпочтение микропроцессорам, которые реализуют математические и логические операции с помощью минимального числа команд. Ссылаясь на табл. 24.2.

заметим, что все основные команды АЛУ относятся к командам обработки данных. Каждое новое поколение микропроцессоров и микроЭВМ характеризуется новыми типами команд обработки данных; первые микропроцессоры выполняли команды, перечисленные в табл. 24.2. В табл. 24.4 и 24.5 приведен расширенный список команд для микропроцессоров второго и третьего поколений. Для реализации некоторых из этих команд, например команд умножения и деления, требуется большое число микрокоманд.

В микропроцессорах третьего поколения реализован специальный класс команд, которые позволяют повторять процесс вычислений до тех пор, пока содержимое счетчика не станет равным нулю. Например, в микропроцессоре Z80 к этому классу команд относится группа команд передачи и поиска. Использование таких команд уменьшает длину программы, но требует для выполнения большего числа шагов. Главное достоинство каждого нового поколения микропроцессоров заключается в расширении возможностей команд обработки данных (для каждой команды предусматривается несколько вариантов ее использования). В то же время непрерывно вводятся новые и улучшаются прежние способы адресации, что позволяет писать более компактные программы. Использование в некоторых микропроцессорах двух или более индексных регистров дает возможность одновременно выполнять несколько операций.

Управление программами. Команды управления данными и пересылки данных обрабатывают преимущественно информацию, хранящуюся в регистрах, аккумуляторах и оперативной памяти. Команды этих двух классов влияют на состояние счетчика команд неявно. Что касается команд управления программой, то они непосредственно изменяют состояние счетчика команд. Эти команды изменяют порядок выполнения команд программы. Команды управления программой могут выполняться либо безусловно, либо по условию. Выполнение **безусловной команды управления** приводит к присваиванию счетчику команд содержимого второго слова (или второго и третьего слов) команды. Выполнение условной команды управления вызывает присваивание счетчику значения второго слова (или второго и третьего слов) команды только в том случае, если выполняются заданные условия. Эти условия определяются либо значениями определенных битов в регистре признаков, либо содержимым какого-нибудь общего регистра.

Как уже отмечалось, с помощью команды перехода осуществляется присваивание счетчику команд значения, определяемого вторым и третьим словами команды. Иначе говоря, если использовать введенную ранее терминологию, применяется

Таблица 24.4. Арифметические команды обработки данных в микропроцессорах второго и третьего поколений

Название	Описание операций
Сложение	Бит переноса не добавляется Бит переноса добавляется Сложение содержимого регистра с содержимым аккумулятора Сложение содержимого двух аккумуляторов Сложение текущего значения с содержимым аккумулятора Сложение содержимого аккумулятора с данным, выбираемым из памяти способом прямой адресации, расширенной адресации, косвенной адресации, адресации с индексированием и адресации по регистру Сложение с многократной точностью Сложение в двоично-десятичном коде
Вычитание	Бит переноса (отрицательный перенос) не учитывается Бит переноса учитывается Вычитание содержимого регистра из содержимого аккумулятора Вычитание содержимого одного аккумулятора из содержимого другого аккумулятора Вычитание текущего значения из содержимого аккумулятора Вычитание из содержимого аккумулятора значения данного, выбираемого из памяти способом прямой адресации, расширенной адресации, косвенной адресации, адресации с индексированием и адресации по регистру Вычитание с многократной точностью
Очистка	Очистка аккумулятора Очистка регистра Очистка области памяти, определенной способом прямой адресации, расширенной адресации и адресации с индексированием
Умножение Деление	Перемножение содержимого двух регистров Деление содержимого пары регистров на содержимое третьего регистра
Увеличение на 1 (Уменьшение на 1)	Увеличение содержимого аккумулятора Увеличение содержимого регистра Увеличение содержимого области памяти, определенной способом расширенной адресации и адресации с индексированием Увеличение указателя стека Увеличение содержимого пары регистров Увеличение содержимого индексного регистра Увеличение и переход, если нуль
Операция отрицания	Получение дополнения к значению аккумулятора и добавление 1

Взято из работы [1].

Таблица 24.5. Логические команды обработки данных
в микропроцессорах второго и третьего поколений

Название	Описание операций
Дополнение Циклический сдвиг	Замена 1 на 0 и 0 на 1 в аккумуляторе Циклический сдвиг вправо, включая перенос Циклический сдвиг влево, включая перенос Циклический сдвиг аккумулятора Циклический сдвиг регистра Циклический сдвиг области памяти, определенной способом расширенной адресации и адресации с индексированием Четырехбитовый циклический сдвиг Циклический сдвиг пары регистров
Сдвиг	Арифметический сдвиг влево, нули помещаются в наименьшие значащие разряды Арифметический сдвиг вправо, наибольшие значащие разряды без изменений Арифметический сдвиг любого аккумулятора Арифметический сдвиг любого регистра Арифметический сдвиг области памяти, определенной способом расширенной адресации и адресации с индексированием Логический сдвиг вправо, в наибольшие значащие разряды помещаются нули
И	Операция И над содержимым аккумулятора и регистра Операция И над содержимым аккумулятора и текущим значением Операция И над содержимым аккумулятора и областью памяти, определенной способом прямой адресации, расширенной адресации и адресации с индексированием
ИЛИ	Операция ИЛИ над содержимым аккумулятора и регистра Операция ИЛИ над содержимым аккумулятора и текущим значением Операция ИЛИ над содержимым аккумулятора и областью памяти, определенной способом прямой адресации, расширенной адресации и адресации с индексированием

Продолжение

Название	Описание операций
ИСКЛЮЧАЮЩЕЕ ИЛИ	Операция ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и регистра Операция ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и текущим значением Операция ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и областью памяти, определенной способом прямой адресации, расширенной адресации и адресации с индексированием
Сравнение	Сравнение содержимого аккумулятора и регистра Сравнение содержимого аккумулятора с текущим значением Сравнение содержимого аккумулятора с содержимым области памяти, определенной способом прямой адресации, расширенной адресации и адресации с индексированием Сравнение содержимого аккумулятора и регистра, уменьшение на 1 содержимого другого регистра

Взято из работы [1].

способ расширенной адресации. Если используется условная команда, то она не выполняется до тех пор, пока условие не удовлетворяется. Выполнение команды перехода по содержимому индексного регистра приводит к загрузке в счетчик команд числа из индексного регистра. Такой способ адресации позволяет динамически менять адрес перехода.

Для команд условного перехода ¹⁾ требуется еще одно дополнительное слово для определения нового значения счетчика команд. В 8-разрядных микропроцессорах это дополнительное слово необходимо для задания адреса в диапазоне: текущее значение счетчика команд ± 127 слов. В этом втором слове, которое часто называют «относительным смещением», для представления смещения используется дополнительный код числа; т. е. если наибольший значащий бит слова равен 1, то смещение считается отрицательным.

При записи команд **CALL** и **JSR** применяется способ расширенной адресации. Их выполнение приводит к тому, что значением счетчика команд становится адрес начала подпрограм-

¹⁾ Эти команды также называют командами ветвления. — Прим. перев.

мы. **RET** и **RTS** — это также команды с расширенной адресацией. Эти команды обязательно должны присутствовать в конце подпрограмм. В результате их выполнения значением счетчика команд становится адрес команды, непосредственно следующей за командой **CALL** или **JSR**, инициировавшей переход к подпрограмме. Во многих микропроцессорах также предусмотрены специальные расширения команд **CALL** и **RET**, позволяющие производить вызов и возврат в зависимости от условия.

24.4. ОСНОВЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

В ближайшем будущем микропроцессоры и микроЭВМ проникнут практически во все сферы человеческой деятельности. В этом разделе приводится описание процесса проектирования нескольких микропроцессорных систем различного применения. Эти области применения достаточно различаются, и в связи с этим большинство тех, кто использует микропроцессоры, найдет этот материал полезным для себя. При написании этой главы, так же, впрочем, как и всей книги, авторы пытались сохранить определенные пропорции при освещении вопросов аппаратного и программного обеспечений.

Ниже приведена пошаговая процедура проектирования микропроцессорных систем. Для одних приложений может потребоваться только несколько шагов этой процедуры, для других — почти все.

ШАГ 1. Перед началом работы над проектом сформулируйте цели разработки и основные требования. Как для больших, так и для малых проектов полезно установить требования к параметрам системы. Это не только важно для проектирования, но и предоставляет руководству и другим заинтересованным лицам возможность оценки проекта и внесения улучшений. Каждый час времени, потраченный на этом шаге, позволит сэкономить сотни часов работы впоследствии.

ШАГ 2. Нарисуйте упрощенную блок-схему аппаратной части с учетом требований к входным данным. Этот шаг позволяет оценить полноту набора требований и указать, следует ли изменить или расширить этот набор. На этом шаге впервые дается приблизительная оценка стоимости проекта. Здесь же могут быть спланированы наиболее крупные и длительные этапы работы над проектом.

ШАГ 3. Нарисуйте упрощенную структурную схему, на которой были бы показаны связи между управляющей программой и основными подпрограммами. Определите специальные подпрограммы, которые должны обеспечить ввод и вывод данных.

На управляющую программу должно быть возложено как можно меньше функций ввода-вывода и вычислительных операций. Управляющая программа должна только связывать различные подпрограммы друг с другом, а вся основная работа должна выполняться подпрограммами.

ШАГ 4. Проведите предварительную оценку скорости вычислений. Учитывая ограничения, налагаемые устройствами ввода-вывода, оцените верхнюю границу быстродействия каждой подпрограммы. Исходя из этого, оцените верхнюю границу времени выполнения полного цикла управляющей программы. Если некоторые или все подпрограммы вызываются с использованием механизма прерывания, то при оценке скорости обработки должны быть учтены времена выполнения каждой подпрограммы обработки прерываний. Если скорость обработки не вызывает опасений, большая часть операций этого шага может быть опущена.

ШАГ 5. Составьте список устройств ввода-вывода с указанием их быстродействия. Эта информация потребуется на следующих шагах. Определите, какими должны быть порты ввода-вывода: параллельными или последовательными, синхронными или асинхронными, с временной синхронизацией или без нее, запирающимися или нет и т. д.

ШАГ 6. Выберите размеры адресного слова и слова данных. Какую длину слова данных должен иметь микропроцессор: 4, 8, 16 и 32 бит? Решение этого вопроса зависит от типов используемых устройств ввода-вывода, требуемой точности и скорости вычислений или от возможностей уже разработанных систем проектирования для данного микропроцессора. Размер адресного слова зависит от требуемого объема памяти.

ШАГ 7. Проведите сравнение наиболее подходящих для проекта микропроцессоров или микроЭВМ. Внимание следует обращать на стоимость, пригодность для реализации проекта, учитывать средства сопровождения программного обеспечения, быстродействие, совместимость с другими типами БИС, способы адресации, характеристики устройств ввода-вывода, оперативной памяти и ПЗУ, оценить систему питания, систему прерываний и набор команд. Часто в распоряжении у проектировщика имеется система проектирования микропроцессоров только одного типа, но такое сравнение все-таки осуществимо, поскольку любая система проектирования обычно обеспечивает разработку программного обеспечения для нескольких одноклассовых процессоров.

ШАГ 8. После определения подходящего микропроцессора выберите вспомогательные кристаллы с большим уровнем интеграции. Убедитесь, что они проверены и снабжены документацией производителя.

ШАГ 9. Разработайте центральный процессор, схемы синхронизации, управления шинами, а также схемы дешифрации адреса. Поскольку эти элементы являются центральными в системе, они должны обладать возможностями, достаточными для управления всей системой, и учитывать перспективы ее роста.

ШАГ 10. Спроектируйте память системы, используя для этих целей ОЗУ, ПЗУ, ППЗУ, перепрограммируемые ПЗУ и т. д. Выберите кристалл микроЭВМ с максимально большим ОЗУ и ПЗУ. Если предполагается крупномасштабный выпуск продукции, выберите кристалл микроЭВМ как с ППЗУ, удобным для проектирования, так и с ПЗУ с масочным программированием, удобным для пользователей.

ШАГ 11. Разработайте схемы ввода-вывода, используя для этого БИС, которые широко применяются и снабжены хорошей документацией. Старайтесь разместить главные шины данных и главные адресные шины внутри самого микропроцессора. Организуя связь с другими платами и устройствами ввода-вывода, используйте вторичные шины данных и декодирующие адресные линии. Многие трудности можно преодолеть, если разместить все вторичные и главные шины на одной плате в пределах некоторой контролируемой области.

ШАГ 12. Разработайте панель управления и другие устройства, служащие для связи с пользователем. Проверьте, все ли средства отладки, предусмотренные в системе, являются экономически обоснованными. Сделав главную панель системы полностью программно-управляемой (как по устройствам ввода, так и по устройствам отображения), следует дополнить аппаратные и программные части системы средствами отладки, которые помогут в дальнейшем сэкономить сотни часов. Эти элементы неocenимы во время разработки системы и помогают при поиске неисправностей.

ШАГ 13. Разработайте систему распределения энергии. Большинство проектировщиков приобретают, а не проектируют источники питания. Убедитесь, что возможности источников питания достаточны для удовлетворения потребностей системы в будущем. Выберите поставщиков, которые предоставят гарантии и будут быстро производить ремонт неисправных блоков. Позаботьтесь о печатных схемах, кабелях и проводниках, необходимых для распределения токов, идущих от каждого источника. Снабдите точными номерами те конденсаторы на платах, которые необходимы для того, чтобы отклонения напряжения в источниках питания не превосходили нескольких милливольт.

ШАГ 14. Теперь, когда накоплено довольно много информации о системе, нарисуйте подробную схему управляющей программы. Составьте список всех подпрограмм. Укажите их типы,

размеры и предполагаемые места размещения в памяти. Используйте концепции модульного программирования. Старайтесь делать подпрограммы в 20—30 операторов. Программы такого размера удобны для отладки и внесения в них изменений как для их составителя, так и для любого, кто будет с ними работать.

ШАГ 15. Задайте максимально эффективное и информативное распределение памяти. Поместите его в начале программного листинга для того, чтобы оно постоянно проходило стадию редактирования вместе с программой.

ШАГ 16. Напишите тексты управляющей программы и всех подпрограмм. Проведите компиляцию, исправление ошибок, повторную компиляцию и подготовьте программу для тестирования. Если возможно, то, используя систему проектирования микропроцессоров, промоделируйте работу аппаратных средств и попытайтесь осуществить прогон программы.

ШАГ 17. Загрузите откомпилированную программу в ОЗУ, ППЗУ или память, имитирующую работу ПЗУ, и проведите тестирование разработанной программой системы. Проведите отладку, повторное проектирование и запишите программу в соответствии с требованиями функционирования системы. На этом этапе работы всегда затрачивается очень много времени. Проекты на этом этапе либо выживают, либо окончательно умирают.

ШАГ 18. В верхней части программного листинга приведите список всех параметров проекта, временных регистров, рабочих регистров, портов ввода-вывода и т. п. Подробно опишите каждый элемент.

ЛИТЕРАТУРА

1. Stout D. F., Kaufman M., Handbook of Microcircuit Design and Application, McGraw-Hill, New York, 1980, pp. 18—20.
2. Nemec J., Lau S. Y., Bipolar Microprocessors: An Introduction to Architecture and Applications, EDN, September 20, 1977, p. 63.
3. Weiss C. D., Software for MOS/LSI Microprocessors, Electronic Design, vol. 7, April 1, 1974, p. 50.
4. Ungermann R., Peuto B., Get Powerful Microprocessor Performance by Using the Z80, Electronic Design, vol. 14, July 5, 1977, p. 54.

К. Виатровски, Ч. Хаус

25.1. ВВЕДЕНИЕ

Понятие микроЭВМ (микрокомпьютер) ассоциируется с очень малой вычислительной машиной. К микроЭВМ принято относить машины, в которых блок центрального процессора реализован в виде одной интегральной схемы. Первые микроЭВМ были действительно очень небольшого размера и обладали незначительной вычислительной мощностью, имели ограничения по объему памяти и совсем малое число связей с внешними устройствами. Область применения таких ЭВМ была весьма ограниченной. Даже эти первые микроЭВМ были значительно дешевле других используемых ЭВМ и имели широкое распространение.

Достижения в области технологии производства интегральных схем резко изменили архитектуру микроЭВМ, поскольку появились устройства, выполненные в виде БИС, которые можно легко подключить к микроЭВМ. Вычислительные мощности таких микропроцессорных систем значительно возросли. Современные микроЭВМ имеют память большого объема и средства связи с разнообразными внешними устройствами. Даже с такими расширениями микроЭВМ стоят значительно дешевле, чем раньше.

В процессе развития становилось все труднее точно выделить характерные черты микроЭВМ. Важными характеристиками продолжали оставаться габариты, стоимость и область применения. Термин микроЭВМ до сих пор почти всегда связывается с ЭВМ, центральный процессор которых реализован в виде одной интегральной схемы или в крайнем случае в виде нескольких интегральных схем. Интегральное исполнение микроЭВМ позволяет существенно снизить их стоимость. Более того, и другие элементы микроЭВМ, такие, как память, теперь размещаются на том же единственном кристалле ИС.

¹⁾ Adapted from Logic Circuits and Microcomputers Systems, by Claude A. Wiatrowski and Charles H. House. Copyright © 1980, Used by permission of McGraw-Hill, Inc. All rights reserved.

Обычно микроЭВМ определенной серии предназначаются для решения какой-то одной задачи или группы взаимосвязанных задач. Раньше для снижения срока окупаемости дорогостоящие ЭВМ обслуживали сразу нескольких пользователей. Низкая стоимость микроЭВМ позволяет использовать их для решения только небольшой группы задач. Поэтому микроЭВМ можно увидеть в автомобилях, швейных машинах, устройствах для завивки волос. Более сложные задачи, связанные с управлением промышленными объектами, разбиваются на отдельные операции, выполнение которых поручается микроЭВМ.

Роль микроЭВМ в жизни общества

Благодаря небольшим размерам и низкой стоимости сфера применения микроЭВМ очень широка. Все большее влияние оказывают микроЭВМ на повседневную жизнь людей. МикроЭВМ помогают увеличить производительность труда и уменьшить стоимость товаров. Кроме того, они позволяют экономить дефицитное сырье и энергию, которая в других случаях была бы израсходована. МикроЭВМ позволяют уменьшить загрязненные среды, контролируя ход производственных процессов.

Они используются в автомобилях для управления моментом зажигания и впрыскиванием горючего, что позволяет максимизировать пройденный путь при сокращении объема выхлопных газов.

Компоненты микроЭВМ

В литературе термины «микропроцессор» и «микроЭВМ» часто используются как синонимы. Хотя разница между этими понятиями все-таки есть, тенденция к размещению всех компонентов ЭВМ на одном кристалле способствует сближению этих понятий. Термин «микропроцессор» иногда используется для обозначения только блока центрального процессора, в то время как термин «микроЭВМ» применяется для обозначения всей системы, состоящей из процессора, памяти и устройств ввода-вывода. В дальнейшем в этой книге термины «микропроцессор» и «микроЭВМ» считаются эквивалентными. На рис. 25.1 приведена схема микроЭВМ. Каждый блок реализуется в виде одной или нескольких интегральных схем и, возможно, включает другие необходимые для работы элементы.

Микропроцессор — главный компонент (ядро) микроЭВМ, который предназначен для управления работой всей системы.

Блок памяти содержит схемы, необходимые для хранения результатов вычислений, выполняемых микропроцессором. В этом блоке также размещается последовательность команд,



Рис. 25.1. Компоненты микроЭВМ.

которая инициирует выполнение вычислений. Важную роль в микропроцессоре играют два типа памяти. Один тип памяти — память, предназначенная для считывания и записи. В этой памяти могут храниться команды программы и данные. Поскольку все данные и команды, содержащиеся в памяти для считывания-записи, уничтожаются, когда питание отключается, необходимо сохранить команды в другой памяти. Для этого может быть использовано внешнее запоминающее устройство, например кассетный магнитофон. Хранение программ на магнитной ленте часто применяется в таких областях использования микроЭВМ, как обработка платежных ведомостей и решение научно-технических задач.

Включение такого периферийного устройства, например, в состав швейной машины с управлением от микроЭВМ было бы слишком обременительно и дорогостояще. И совершенно невозможно использовать такое устройство в системе управления автомобиля. Все это приводит к необходимости применять для долгосрочного хранения программ и данных специальный тип памяти, предназначенной только для чтения. Такое устройство памяти называется **постоянным запоминающим устройством (ПЗУ)**. Термин «ЗУ с произвольной выборкой» применим к обоим перечисленным выше типам памяти. Любая совокупность данных, хранящихся в этой памяти, может быть вызвана с помощью микропроцессора. Однако исторически сложилось так, что термин ЗУ с произвольной выборкой почти всегда относится к памяти, предназначенной для считывания-записи,

Внешние устройства связываются с микропроцессором с помощью специальных средств сопряжения — интерфейсов. Интерфейсы зачастую бывают узкоспециализированными. Например, интерфейс для печатающего устройства непригоден для организации связи микроЭВМ с тормозными цилиндрами железнодорожной платформы. Большинство аппаратных средств, необходимых для эксплуатации микроЭВМ, представляют собой схемы интерфейсов. Примечательно то, что многие интерфейсы выполняют несколько одинаковых функций. Поэтому изготовители микроЭВМ обычно отдают предпочтение одной схеме, которую можно настраивать. Такие схемы получили название программируемых интерфейсов ввода-вывода. Они предназначены для часто используемых периферийных устройств, таких, как устройства печати, дисплеи, магнитные диски и устройства ввода с клавиатуры. Интерфейсы ввода-вывода будут подробно рассмотрены в следующей главе.

Схема синхронизации обеспечивает согласование работы всей микропроцессорной системы. Схема установки в исходное состояние необходима для инициализации работы процессора после включения питания. Обе эти схемы в появившихся в последнее время микропроцессорных системах обычно размещаются на том же кристалле, что и микропроцессор. Источники питания необходимы для обеспечения напряжением всех устройств системы.

Роль программирования

Хотя распространение микроЭВМ способствовало увеличению областей применения вычислительной техники, неочевидно, что быстрый рост числа ЭВМ несет только пользу. Применение ЭВМ может быть выгодным только в том случае, если их удастся запрограммировать для решения конкретных задач. Программой называется последовательность команд для центрального процессора, выполнение которых позволяет решить данную задачу. Составление такой последовательности команд называется программированием. Совокупность программ часто называют программным обеспечением (software) для того, чтобы не путать с логическими схемами, называемыми аппаратным обеспечением (hardware). Для того чтобы использовать микроЭВМ для решения конкретной задачи, необходимо разработать программу и систему интерфейсов. Все остальное не зависит от области применения. Поскольку эти оставшиеся компоненты (процессор, память и т. д.) являются универсальными, их экономически выгодно производить массовым тиражом.

Функции, выполняемые микроЭВМ, могут быть изменены путем замены программы, хранящейся в ПЗУ. Такая модифи-

кация является относительно простым делом. Она бывает необходимой для настройки системы на конкретное применение или просто для исправления первоначально допущенной ошибки.

Дополнительные программы позволяют придать системе дополнительные функции, что приводит к незначительному возрастанию ее стоимости. Даже если требуется дополнительное ПЗУ, то средства, затраченные на его установку, окупятся программами, которые пользователи разместят в нем.

Разработка программ — основная часть процесса проектирования микропроцессорной системы. Поэтому программирование и является первой темой в этой главе.

25.2. ОПИСАНИЕ ПРОГРАММ

Поскольку программа реализует некоторый алгоритм, ее структуру можно описать по аналогии с описанием алгоритма. Наиболее распространенным является словесное описание того, что программа делает. Часто именно со словесного описания следует начинать разработку программы. Однако словесное описание становится очень громоздким, когда дело касается реализации и тестирования программы. Часто для этого используются схемы программ, хотя существуют и другие методы, от которых, если они проще, не следует отказываться. На практике любые полезные приемы могут и должны быть использованы для упрощения разработки.

Схемы, которые в этой книге применяются для описания программ, проще схем, которые служат для описания алгоритмов. На рис. 25.2 показан символ «Процесс», соответствующий

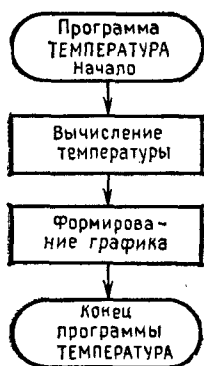


Рис. 25.2. Символы «Пуск-останов» и «Процесс».

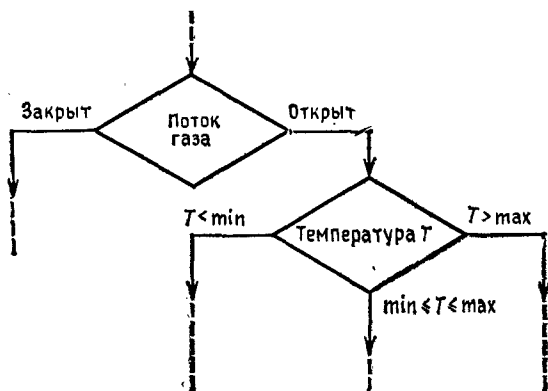


Рис. 25.3. Символы «Решение».

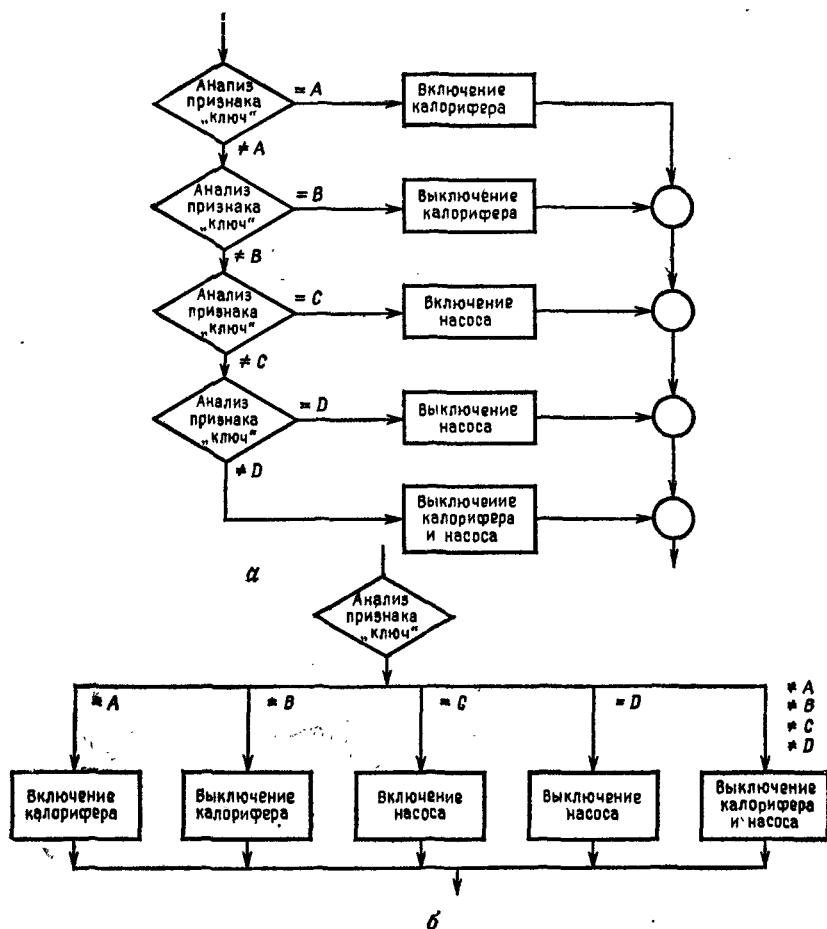
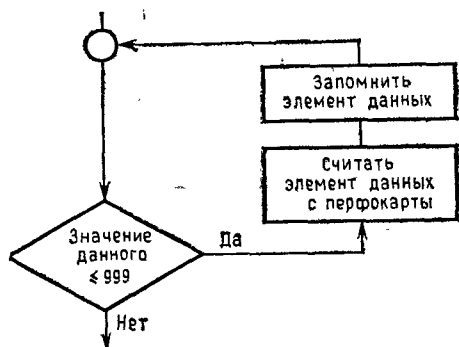
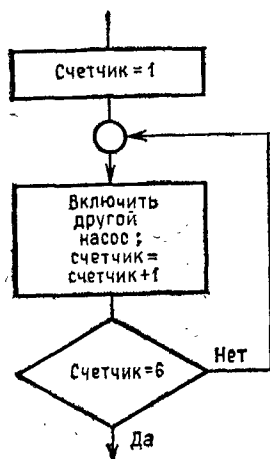


Рис. 25.4. Множественное ветвление.

а — последовательная проверка условий; б — одновременная проверка условий.

блоку обработки, и символ «Пуск-останов», применяемый для обозначения начала, окончания или прерывания процесса обработки данных. Использование последнего символа позволяет упростить изображение схемы, если она размещается на нескольких листах. На каждом листе можно располагать по одной программе, а связи между ними устанавливать с помощью символа «Пуск-останов».

Символы «Решение», соответствующие блокам принятия решений, могут иметь более двух выходов (рис. 25.3). На этом рисунке приведена схема, позволяющая установить, в какой из

Рис. 25.5. Схема конструкции **do while**.Рис. 25.6. Схема конструкции **do until**.

трех диапазонов попадает значение температуры. На рис. 25.4 показаны два способа изображения множественного ветвления. На рис. 25.4, а изображена схема, предназначенная для последовательной проверки условий ветвления. В этом случае переменная **KEY** сравнивается сначала с **A**, затем с **B** и т. д.

На рис. 25.4, б изображена схема, в которой все условия проверяются одновременно. В этом случае необходимо, чтобы все возможные варианты были учтены. Обычно в схемах, описывающих одновременное принятие решений, не совсем точно отражено то действие, которое необходимо выполнить при проверке всех входных условий. Следует отметить, что микроЭВМ могут выполнять операции проверки условий как последовательно, так и одновременно. Поэтому следует использовать ту форму схем, которая в данном конкретном случае реализуется.

Каждому алгоритму соответствует практически неограниченное число структурных схем. Для того чтобы уменьшить вероятность появления ошибок, при построении структурных схем следует использовать только несколько конструкций. В этой книге для изображения последовательности действий один блок обработки будет размещаться вслед за другим и соединяться с ним стрелкой. При описании схем, содержащих блоки принятия решений, рекомендуется использовать две конструкции — «ветвление» и «цикл». На рис. 25.4 показаны схемы, соответствующие конструкции «ветвление». Здесь блоки обработки выполняются или не выполняются в зависимости от значений переменной **KEY** (ключ).

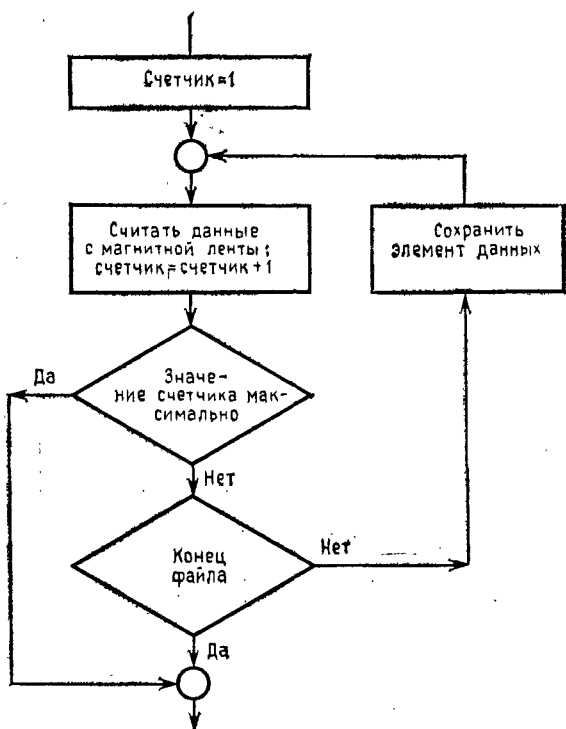


Рис. 25.7. Схема, содержащая разные типы циклов.

На рис. 25.5, 25.6 показаны схемы циклов. В соответствии со схемой рис. 25.5 данные считываются с перфокарт, последовательно вводимых с помощью системного устройства ввода. Этот процесс оканчивается, как только появится элемент данных, больший 999. Такая конструкция называется циклом **do while**, поскольку одни и те же действия повторяются, пока некоторое условие остается истинным. Конструкции **do while** часто используются в тех случаях, когда число итераций, как и в данном примере, заранее не определено.

На рис. 25.6 показана схема конструкции **do until**. Эта структурная схема предусматривает вступление в работу шести насосов. Аналогичные действия повторяются до тех пор, пока не станет истинным некоторое условие. В данном примере этим условием является вступление в работу шестого насоса. Конструкции **do until** обычно используются тогда, когда число выполняемых действий известно заранее. В действительности обе конструкции **do until** и **do while** могут быть использованы как

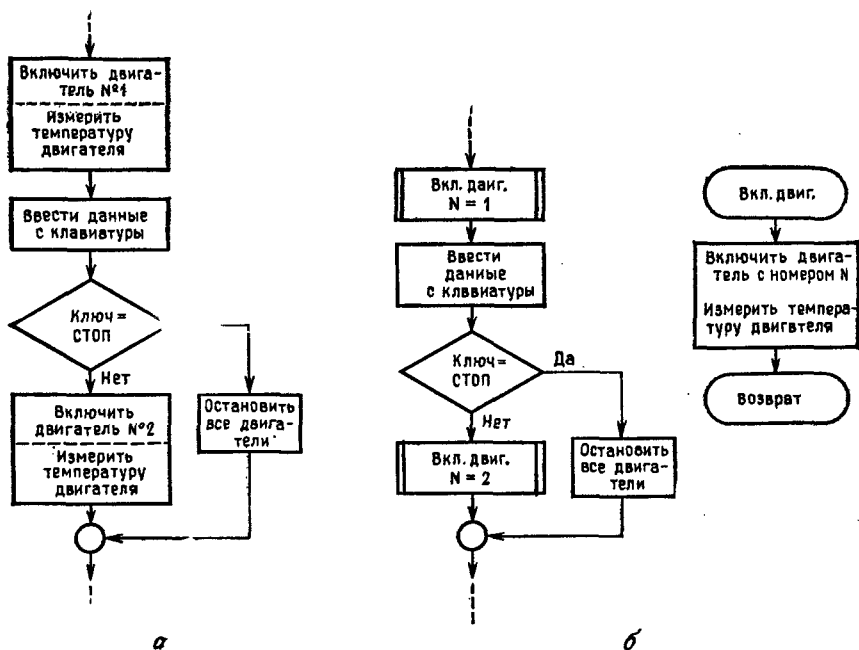


Рис. 25.8. Упрощение схемы.

а — схема с повторяющимися действиями; б — упрощенная схема.

в случае определенного, так и в случае неопределенного числа итераций.

В каждом конкретном случае должен быть выбран тип цикла, наиболее точно удовлетворяющий условиям задачи. Но все же между этими двумя типами циклов существует важное отличие, которое проявляется, когда условие истинно перед входом в цикл. В этом случае в цикле **do until** предусмотрено выполнение действий один раз, а в цикле **do while** действия не выполняются.

Можно объединить циклы обоих типов в рамках одной схемы (рис. 25.7). Многие действия на ЭВМ включают ввод элемента данных, проверку, не является ли этот элемент последним, и его обработку. Эта последовательность действий повторяется для всех данных в списке.

На рис. 25.8 изображены две схемы, описывающие выполнение одних и тех же действий. Заметим, что два блока обработки на рис. 25.8, а весьма похожи. Упростим схему, записав общие для двух блоков обработки процессы отдельно, и будем выполнять обращение к ним из главной схемы, когда это

необходимо (рис. 25.8, б). Такие общие для нескольких блоков процессы могут быть более сложными и могут включать много блоков обработки и принятия решений. По поводу схемы на рис. 25.8, б заметим все же, что, перед тем как обращаться к последовательности действий **Вкл. двиг.**, необходимо определить, какой двигатель надо запустить. Номер двигателя *N* называется **аргументом**. В следующем параграфе будет показано, что использованное выше упрощение изображения не является просто формальным приемом, а может быть реализовано в микроЭВМ в виде подпрограммы.

25.3. ТИПЫ ДАННЫХ

Выполнение большинства команд микропроцессора связано с операциями над данными. В микроЭВМ для упрощения составления программ используется несколько стандартных типов данных. Без сомнения, основным элементом данных остается бит, который может принимать два значения: либо 0, либо 1. Сцепление восьми битов образует **байт**. Байт — это наиболее распространенный элемент данных для микроЭВМ. Иногда цепочку из восьми или более битов называют **словом**. Обычно длина слова равна 16 бит.

Совокупность битов можно использовать для представления многих типов данных. Простейшая микроЭВМ, которая в этой книге будет фигурировать в качестве примера, допускает только несколько представлений. На рис. 25.9 показано представление двоичного целого без знака. Каждый бит задает некоторую степень числа 2. Если бит равен 1, то соответствующая степень числа 2 присутствует в искомом числе. С помощью 8 бит могут быть представлены положительные числа от 0 до 255. Двоичная точка не обязательно должна быть размещена справа от числа, она может быть в любом месте (рис. 25.10). Хотя числа, больше 255, представить уже не удастся, зато есть возможность представлять дробные значения. Позиция двоичной точки не хранится вместе с данными; она обычно существует только в мыслях программиста и в том значении, которое он вкладывает в каждый бит.

Ранее уже говорилось, что дополнительный код наиболее часто используется для представления двоичных чисел со знаком. На рис. 25.11 показано, какие веса связываются с каждым битом в 8-битовом дополнительном коде числа. Если наибольший значащий бит равен 1, то число является отрицательным. Такое взвешивание отрицательных чисел приводит к двоичному представлению, показанному на рис. 25.12. На нем изображены дополнительные коды 8-битовых целых чисел. Значения чисел лежат в диапазоне от -128 до $+127$. Если первый бит равен 1,

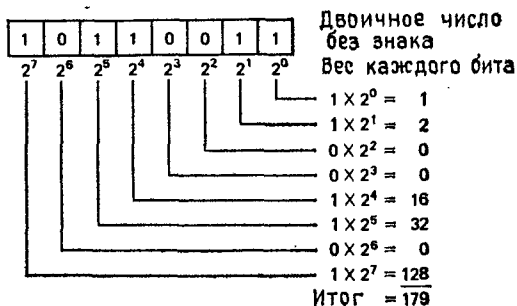


Рис. 25.9. Представление числа 179 в виде двоичного целого без знака.

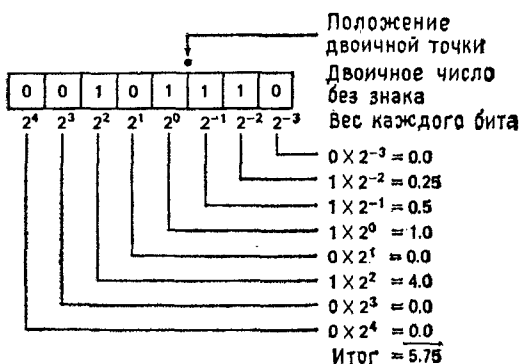


Рис. 25.10. Представление числа 5,75 в виде двоичного числа без знака.

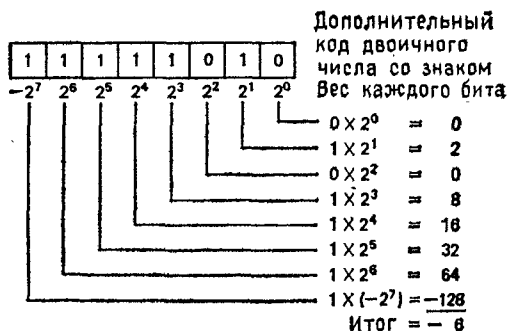


Рис. 25.11. Представление числа (-6) в дополнительном двоичном коде.

Десятичное число Двоичное число

127	0111	1111
126	0111	1110
2	0000	0010
1	0000	0001
0	0000	0000
-1	1111	1111
-2	1111	1110
~127	1000	0001
~128	1000	0000

Рис. 25.12. Дополнительные коды целых чисел.

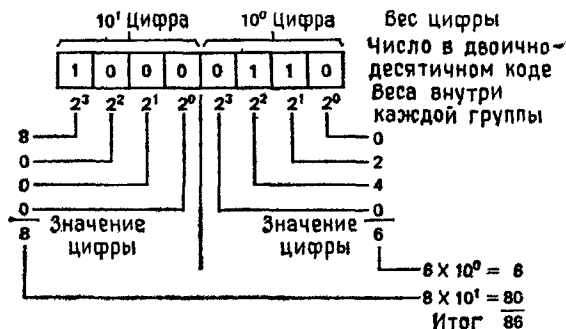


Рис. 25.13. Представление числа 86 в двоично-десятичном коде.

то задается отрицательное число. Если первый бит равен 0, число — положительное. Советуем читателю хорошо усвоить представление двоичных чисел в дополнительном коде. Микро-ЭВМ переводит отрицательные числа в дополнительный код, изменяя все биты числа на противоположные (т. е. формируя обратный код) и добавляя 1.

Как подчеркивалось выше, двоичное представление является наиболее эффективным как для хранения чисел, так и для проведения над ними операций. Однако преобразование двоичных чисел в десятичные и, наоборот, десятичных в двоичные осуществить достаточно сложно; а все входные и выходные данные воспринимаются человеком только в десятичном виде. Если нет серьезных ограничений по объему памяти и скорости вычислений, то можно использовать двоично-десятичный код (BCD). Данные, представленные в этом коде, легко преобразуются в форму, удобную для восприятия человеком. На рис. 25.13 показано представление числа в 8-разрядном двоично-десятичном коде. Двоично-десятичные коды чисел делятся на 4-битовые группы. Биты внутри каждой группы являются обычными взвешенными двоичными разрядами, но с их помощью разрешается представлять только числа от 0 до 9. Каждая группа или цифра в двоично-десятичном коде имеет вес, равный определенной степени числа 10. Отметим, что с помощью 8-разрядного двоично-десятичного кода можно представлять числа от 0 до 99, в то время как с помощью обычного

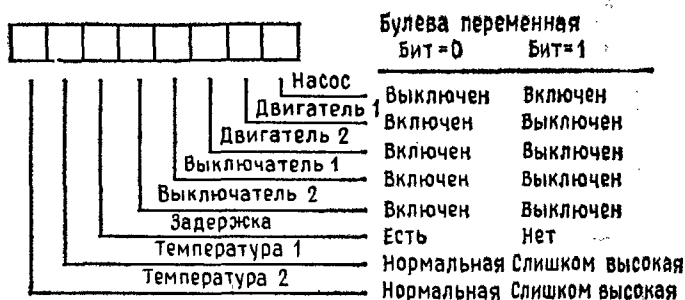


Рис. 25.14. Булевы, или логические, переменные.

8-разрядного двоичного кода можно представлять числа от 0 до 255.

Очень часто для представления требуемого элемента данных бывает достаточно одного бита. Двигатель может быть включен или выключен, дверь открыта или закрыта. В этих двух случаях одного бита будет достаточно для описания состояния. Например, если бит равен 1, то двигатель включен, если бит равен 0, то выключен. Поскольку байт состоит из 8 бит, то его можно использовать для представления данных о восьми состояниях (рис. 25.14). Такие биты естественно связать с булевыми или логическими переменными.

25.4. ХРАНИЕНИЕ ДАННЫХ

Данные могут храниться и извлекаться либо из регистров, либо из памяти. Регистры размещены непосредственно в микропроцессоре, а для памяти выделен отдельный блок (рис. 25.1). Каждый регистр или элемент памяти содержит один байт данных. С точки зрения программиста, основное отличие между памятью и регистрами в том, что регистров значительно меньше, чем элементов памяти. Кроме того, выполнение операций над данными из регистров осуществляется быстрее, чем выполнение тех же операций над данными, извлекаемыми из памяти. Регистры обычно используются для временного хранения промежуточных результатов, в то время как память — для длительного хранения результатов вычислений. В микропроцессорах, имеющих восемь регистров, три бита команды используются для указания того, какой именно регистр принимает участие в операции. Не так уж редко в микропроцессорах применяется память объемом до нескольких десятков тысяч байт. Для представления чисел от 0 до 65 535 нужно 16 бит. Каждое целое число из этого диапазона определяет элемент памяти, а эти 16 бит называются **адресом**. Каждый элемент памяти имеет уникальный адрес, что позволяет микропроцессору выбрать

именно те 8 бит памяти, которые будут использованы в данной операции.

Кроме памяти и регистров микропроцессоров, данные могут размещаться в регистрах интерфейсов с периферийными устройствами. Эти регистры нужны для передачи данных в микропроцессорную систему и из нее. Каждый регистр интерфейса имеет **уникальный адрес**, позволяющий микропроцессору определить, какое именно периферийное устройство нуждается в обслуживании.

25.5. КОМАНДЫ

Для того чтобы микропроцессор мог выполнить ту или иную операцию, необходимы инструкции, или команды. Эти команды хранятся в основной памяти. Каждая команда — это группа битов, соответствующая определенной операции. Обычно команда делится на поля, каждое из которых определяет какой-либо атрибут команды (рис. 25.15). Поле кода операции определяет операцию, которая должна быть выполнена, например

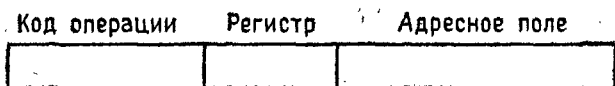


Рис. 25.15. Разделение команды на поля.

сложение, вычитание, пересылка, дополнение и т. п. Код каждой операции представляет собой уникальную комбинацию битов. Поле регистра определяет номер регистра, который содержит данные, принимающие участие в операции; адресное поле служит для определения участка памяти, содержащего данные, используемые при вычислениях.

Команды размещаются в памяти в той последовательности, в которой они должны быть выполнены. Очередная выполняемая команда определяется с помощью содержимого специального регистра, называемого **счетчиком команд**. После выполнения каждой команды, в счетчик команд добавляется единица. Счетчик команд содержит адрес области памяти, в которой хранится следующая, подлежащая выполнению команда. Микропроцессор использует содержимое счетчика команд для выборки из памяти этой команды. Во время выполнения команда временно хранится в специальном регистре микропроцессора — регистре команд.

Различные команды могут потребовать для размещения разного числа битов. Например, в некоторых командах присутствует адресное поле, а в других нет. Так как задавать длину всех команд максимальной нерационально, большинство микроЭВМ использует команды переменной длины. Один

команды занимают 1 байт, другие — 2 байт, а некоторые — 3 байт. Поэтому для поиска адреса следующей команды микропроцессор должен добавить несколько единиц (1, 2 или 3) к содержимому счетчика команд в зависимости от длины последней выполненной команды.

25.6. ОРГАНИЗАЦИЯ МИКРОЭВМ НА БАЗЕ МИКРОПРОЦЕССОРОВ 8080/8085 ФИРМЫ Intel

Микропроцессоры серии 8080/8085 фирмы Intel являются простейшими 8-битовыми микропроцессорами. Приступим к их изучению. С точки зрения программиста эти микропроцессоры включают:

1. Семь рабочих регистров по 8 бит каждый.
2. Память объемом 65 536 байт.
3. Счетчик команд на 16-разрядном регистре.
4. Указатель стека на 16-разрядном регистре.
5. Регистр флажков.
6. Интерфейс ввода-вывода, включающий 256 входных и 256 выходных регистров по одному байту каждый.

Каждый рабочий регистр обозначается определенным числом либо буквой, как показано ниже

Имя	Число	Буква
Аккумулятор	7	A
Регистр	0	B
Регистр	1	C
Регистр	2	D
Регистр	3	E
Регистр	4	H
Регистр	5	L

Регистры иногда используются парами для того, чтобы можно было обращаться к 16-битовым операндам или адресам. К паре регистров, или регистровой паре, можно обратиться, задавая либо число, либо букву, следующим образом:

Имя	Число	Буква	Запрашиваемые регистры
Регистровая пара	0	B	B и C
Регистровая пара	1	D	D и E
Регистровая пара	2	H	H и L
Слово состояния программы	3	PSW	A и флажки
Указатель стека	3	SP	Указатель стека

Регистр флажков

Этот регистр содержит пять булевых переменных, называемых флажками состояния. Они приведены ниже

Номер бита	Буква	Название
7	S	Знак
6	Z	Нуль
4	A	Добавочный перенос
2	P	Четность
0	C	Перенос

Эти флажки изменяются в результате выполнения команд. Для удобства каждый флаг имеет стандартную интерпретацию, в соответствии с которой действуют большинство команд.

Z-бит, или бит нуля, устанавливается в 1, если результат действия команды равен 0. В противном случае он устанавливается в 0.

S-бит, или бит знака, устанавливается в 1, если наибольший значащий бит результата равен 1. В противном случае он устанавливается в 0.

P-бит, или бит четности, устанавливается в 1, если результат содержит четное число единиц.

C-бит, или бит переноса, устанавливается в 1, если после выполнения арифметической операции генерируется либо перенос, либо заем. Роль этого бита при выполнении арифметических операций будет обсуждаться позднее. Бит переноса используется также для других целей при выполнении неарифметических операций.

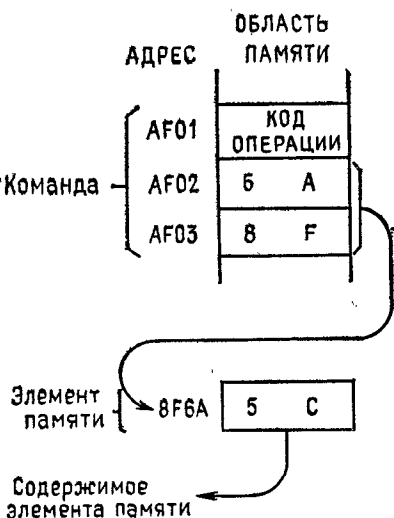
A-бит, или бит добавочного переноса, устанавливается в 1, если при выполнении операции десятичной арифметики формируется перенос из третьего бита в четвертый. Подробнее об этом будет говориться ниже.

Способы адресации

Производительность любой ЭВМ во многом зависит от того, как данные хранятся в памяти и выбираются оттуда. МикроЭВМ 8085 для обращения к данным, хранящимся в памяти, должна сформировать 16-битовый адрес. Для задания адреса в микропроцессоре 8085 фирмы Intel предусмотрено четыре различных способа адресации. Они носят следующие названия: прямая адресация, адресация по паре регистров, адресация с помощью указателя стека, непосредственная адресация.

Прямая адресация

На рис. 25.16 показано, как реализуется простейший способ адресации — прямая адресация. Каждая команда, использующая прямую адресацию, содержит полный адрес той области памяти, где находятся используемые данные. Для представления такой команды требуется три байта. Первый байт содержит код операции, второй — младший значащий байт адреса, третий — старший значащий байт адреса.



Адресация по паре регистров

Можно задать адрес области памяти, используя содержимое регистровой пары, что соответствует 16-битовому слову. На рис. 25.17 показано, как в этом случае происходит обращение к определенному элементу памяти. Адрес области памяти, к которой надо обращаться несколько раз, хранится в регистровой паре. Поэтому в командах, использующих этот адрес, нет необходимости в двух дополнительных байтах. Таким образом,

Содержимое элемента памяти
Рис. 25.16. Прямая адресация.

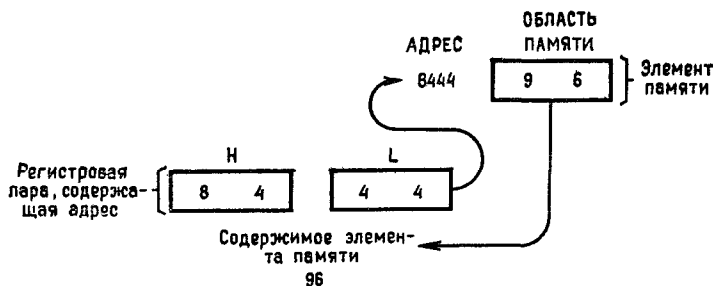


Рис. 25.17. Адресация с помощью регистровой пары.

экономится память, а скорость выполнения увеличивается, поскольку не затрачивается время на считывание из памяти дополнительных байтов. Чаще всего этот способ адресации используется для работы со списками данных, поскольку в микропроцессорах реализованы специальные операции, которые производят уменьшение или увеличение на 1 содержимого регистровой пары, что дает возможность легко перейти к следующему элементу последовательного списка данных.

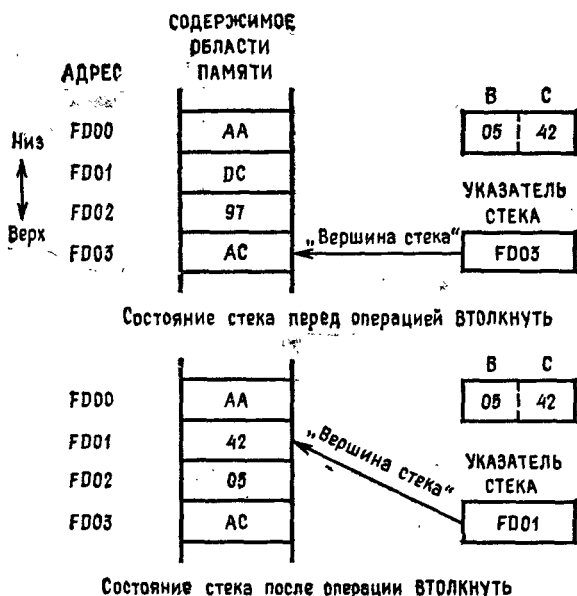


Рис. 25.18. Схема выполнения операции ВТОЛКНУТЬ содержимое регистровой пары В в стек.

Обычно для адресации по паре регистров используется регистровая пара **Н**. В некоторых случаях допустимо применять регистровые пары **В** и **Д**.

Адресация с помощью указателя стека

Этот способ адресации позволяет программисту автоматически производить добавление или удаление 16-битовых элементов данных из списка. Такой список носит название стека и хранится в ЗУ с произвольной выборкой. Со **стеком** можно производить только две операции — **ВТОЛКНУТЬ (PUSH)** и **ВЫТОЛКНУТЬ (POP)**. Выполнение операции **ВТОЛКНУТЬ** происходит следующим образом. Содержимое регистровой пары передается в стек так: первые 8 бит помещаются в область памяти, адрес которой на 1 меньше значения указателя стека (**УС**), вторые 8 бит помещаются в область памяти, адрес которой на 2 меньше значения **УС**. В регистр, содержащий указатель стека, помещается значение, на 2 меньшее предыдущего. Это делается для того, чтобы подготовить стек к размещению следующих 16 бит информации. Весь процесс работы со стеком показан на рис. 25.18. Указатель стека всегда устанавливается в положение, указывающее на последний помещенный в стек байт информации, который называется **вершиной стека**.

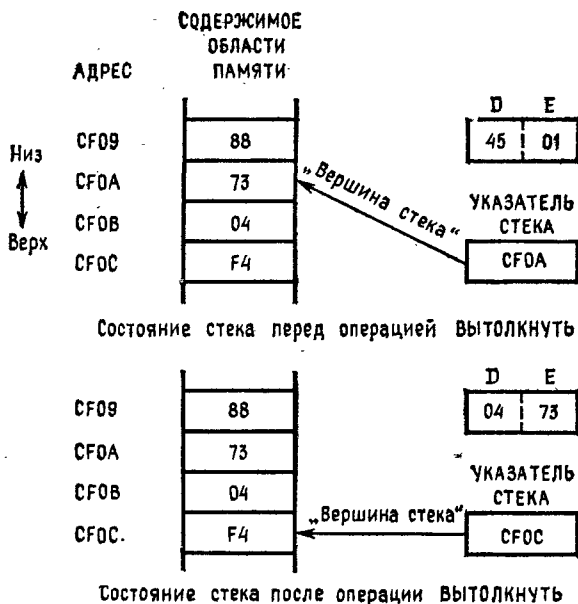


Рис. 25.19. Схема выполнения операции ВЫТОЛКНУТЬ данные в регистровую пару.

Операция **ВЫТОЛКНУТЬ** выбирает из стека 16 бит информации. Первые 8 бит считываются из памяти, адрес которой определяется значением **УС**; вторые 8 бит берутся из памяти, адрес которой на 1 больше значения **УС**. Затем к первоначальному значению **УС** добавляется 2. Это делается для того, чтобы установить **УС** в положение, указывающее на следующий элемент данных. После этого стек считается подготовленным для выполнения очередных операций **ВЫТОЛКНУТЬ** и **ВТОЛКНУТЬ**. Действия, проводимые в соответствии с операцией **ВЫТОЛКНУТЬ**, иллюстрируются на рис. 25.19.

Стек можно определить как список переменной длины, который расширяется книзу (в сторону старших адресов), когда к нему добавляются данные (операция **ВТОЛКНУТЬ**), и который сжимается сверху (со стороны младших адресов), когда данные из него выбираются (операция **ВЫТОЛКНУТЬ**). Важно отметить, что стеки используются для реализации подпрограмм. Этот вопрос будет обсуждаться позднее.

Непосредственная адресация

Когда используется способ непосредственной адресации, реальные данные помещаются непосредственно в саму команду,

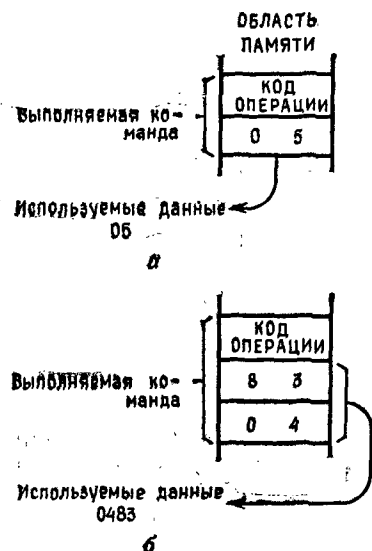


Рис. 25.20. Непосредственная адресация.

а — с элементом данных длиной 1 байт, б — с элементом данных длиной 2 байт.

как показано на рис. 25.20. Элемент данных в этом случае может иметь длину, равную 8 или 16 бит. Тогда в первом случае необходима двухбайтовая команда; один байт из этих двух используется для хранения 8-битового элемента данных. В случае 16-битового данного нужна трехбайтовая команда; два байта используются для хранения 16-битового элемента данных. Непосредственная адресация дает возможность работать сразу с реальными данными. 16-битовый операнд может быть и адресом памяти для регистровой пары **H**, чтобы, например, реализовать адресацию по паре регистров.

25.7. ЯЗЫК АССЕМБЛЕРА

Команды микропроцессора — это на самом деле, последовательности нулей и единиц, представляющие действия, которые необходимо выполнить. Раньше было показано, как двоичные данные можно представить в шестнадцатеричной системе счисления. Ясно, что шестнадцатеричная система может быть использована и для описания команд микропроцессора. Но программы, состоящие из таких команд, будут достаточно трудными для понимания. Для того чтобы облегчить понимание программ, их обычно пишут на языке ассемблера. На этом языке комбинации битов записываются обычно с помощью имен или мнемонических сокращений, которые соответствуют операциям команды. Например, команда микропроцессора 8085 фирмы Intel, заключающаяся в пересылке 8 бит из регистра **A** в регистр **D**, представляется следующим образом в двоичном коде:

0101 0111

или в шестнадцатеричном коде

57H.

Эта команда записана на машинном языке, поскольку именно такое представление используется в микропроцессоре. Она

состоит из трех полей — поля операции, поля регистра назначения и поля регистра источника (рис. 25.21). Теперь выразим имена этих трех полей мнемонически так, чтобы было понятно программисту. Поскольку поле операции описывает пересылку



Рис. 25.21. Машинный язык и язык ассемблера.

(**MOVE**) данных, назовем эту операцию **MOV**. Аналогично определим содержимое поля назначения символом **D** и поля источника символом **A**, соответственно для регистров **D** и **A**. Тогда запись этой команды на языке ассемблера имеет вид

MOV D, A

Такая команда гораздо проще для восприятия, чем аналогичная команда на машинном языке. Поэтому программы очень часто пишутся на языке ассемблера. После того как программа написана на ассемблере, ее необходимо преобразовать в программу на машинном языке либо вручную (ручное ассемблирование), либо автоматически с помощью специальной программы ЭВМ (такие программы называются ассемблерами).

Кроме того, хотелось бы иметь возможность обращаться к элементам памяти с помощью меток, представляющих адреса таких позиций, а не с помощью громоздких шестнадцатеричных чисел. Так, к области памяти, содержащей значение температуры двигателя 1, удобно сослаться по метке **TEMP1**; а к первой строке программы — по ее метке, скажем, **START**. Такой фрагмент программы имеет вид

START: MOV D, A
MOV B, C
 ⋮ ⋮

Однако не следует помечать все элементы памяти. Метки надо присваивать только тем элементам, к которым ссылаются команды программы. Считается, что непомеченные элементы непосредственно следуют за помеченными. Так, команда **MOV B, C** располагается в позиции с адресом **3C11H**, если адрес метки **START** **3C10H**.

Предположим, что программу, написанную на ассемблере, необходимо автоматически **ассемблировать** (т. е. перевести на машинный язык). Тогда необходимо сообщить ассемблеру, каков адрес *первой* метки программы, или, если программа не занимает непрерывного участка памяти, надо указать ассемблеру несколько адресов. Например, можно следующим образом определить, что **START** займет позицию памяти с адресом **3C10H**:

```
ORG 3C10H
START: MOV D, A
      MOV B, C
```

Аббревиатура **ORG** используется для определения начального адреса. Следующая команда программы после *псевдокоманды* **ORG** размещается в области памяти с адресом, определенным в **ORG**. В данном случае следующая команда имеет метку **START** и адрес **3C10H**. Следующая после **ORG** команда программы необязательно должна иметь имя для того, чтобы ей можно было присвоить адрес, определенный в псевдокоманде **ORG**. В машинном языке псевдокоманд нет. Они используются только в программах на языке ассемблера и при ручном ассемблировании с целью явно указать, как программа должна переводиться на машинный язык (например, каким должен быть начальный адрес).

Использование меток не только делает программу более удобной для понимания, но и имеет другие преимущества. Очень часто, для того чтобы исправить ошибку в программе или снабдить программу дополнительными возможностями, требуется между какими-то двумя командами программы вставить еще несколько команд. Все команды, расположенные за этими вновь введенными командами, тогда необходимо передвинуть, чтобы освободить место для новых команд. Шестнадцатеричные адреса *всех* передвигаемых команд должны быть изменены. В соответствии с их новыми адресами должны быть изменены и все *ссылки* к этим командам, т. е. программа на машинном языке должна претерпеть существенные изменения.

Однако, поскольку *метки* изменять не надо, программа на языке ассемблера останется прежней. Ассемблер только по-другому свяжет метки с адресами памяти, чтобы освободить место для новых команд. Например, предположим, что первоначальная версия программы изображена на рис. 25.22, а. В данном случае нет необходимости знать, что делает каждая команда; отметим только, что две команды ссылаются к остальным, используя их метки **START** и **NEXT**. Допустим, что необходимо дополнить программу одной командой. Вновь полученная программа изображена на рис. 25.22, б. Отметим, что

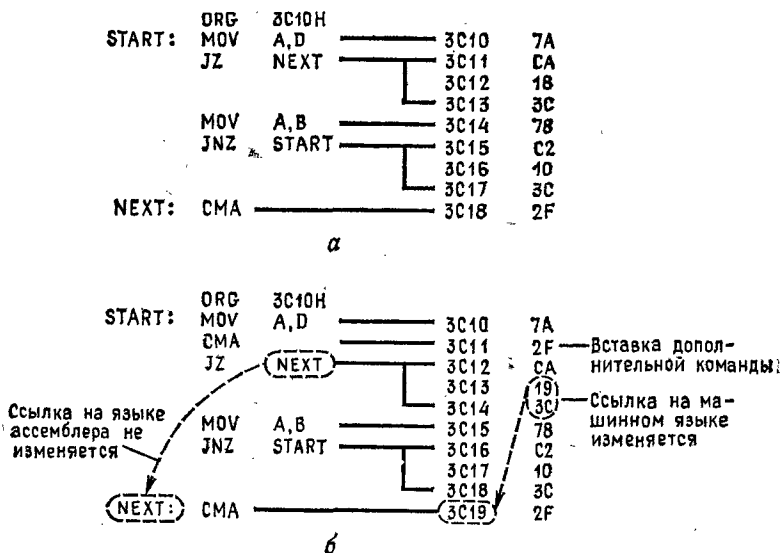


Рис. 25.22. Программа.

а — первоначальная версия; б — модифицированный вариант.

ни одна метка первоначальной программы не была изменена. Однако ассемблер *назначит* новые адреса этим меткам, чтобы освободить пространство для новой команды. Ассемблер также автоматически изменит все ссылки к измененным элементам памяти, соответствующим этим меткам. Ассемблер, таким образом, производит всю огромную рутинную работу, связанную с изменением ссылок при модификации программы.

Для удобства программа на языке ассемблера разделяется на поля. Как это делается, показано ниже:

Метка	Код	Операнд	Комментарий
HERE:	MVI	C, 56H	;ЗАГРУЗКА C ПО АДРЕСУ 56H
THERE:	JMP	NEXT	;ПЕРЕХОД К NEXT
MAYBE:	XRA	D	;ОПЕРАЦИЯ ИСКЛЮЧАЮЩЕЕ ИЛИ МЕЖДУ АККУМУЛЯТО- РОМ И РЕГИСТРОМ D
	CMA		;ДОПОЛНЕНИЕ АККУМУЛЯТО- РА

Поле «метка» может содержать любое символическое имя, ссылающееся к той позиции памяти, которой впоследствии будет

назначен адрес. За каждым именем обязательно следует двоеточие (:). Поле «код» содержит обозначение той операции, которую необходимо выполнить. Поле источника и поле назначения определяются в поле «операнд». Это может быть буква, обозначающая регистр, непосредственные данные или символическое имя. Если требуется задать оба операнда, то они записываются в поле «операнд» через запятую (,). Дополнительным и в то же время очень важным является поле «комментарий». Это поле отделяется от других полей точкой с запятой (;). Все пояснения, касающиеся работы программы, заносятся программистом именно в это поле. Описанные выше поля языка ассемблера относятся только к конкретному ассемблеру микропроцессора 8085 фирмы Intel. Другие ассемблеры могут интерпретировать поля по-другому, хотя в принципе все ассемблеры схожи. Наиболее часто встречаются следующие отклонения: после метки не ставится двоеточие, и комментарий отделяется от остальных полей не точкой с запятой, а каким-либо другим символом.

25.8. КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

Приступим к изучению команд микропроцессора 8085 фирмы Intel. Разделим их на группы в соответствии с теми функциями, которые они выполняют. Первой будем изучать группу команд, предназначенных для простой пересылки данных. Данные можно пересылать между памятью и регистрами. В дальнейшем каждая команда или группа аналогичных команд будет представляться в некотором стандартизованном формате. Каждый раздел начинается с двоичного представления команды. Затем приводится запись команды в формате, принятом для языка ассемблера. И наконец, перечислены биты регистра признаков, которые могут изменяться при выполнении команды. Для простоты при описании команд использованы следующие сокращения:

ADDR.	Имя области памяти
MSPA.	Старший байт адресного поля
LSPA.	Младший байт адресного поля
(LABEL:).	Метка
DATA.	Имя или шестнадцатеричное представление элемента данных
MSPD.	Старший байт 16-битового элемента данных
LSPD.	Младший байт 16-битового элемента данных
DATA 8.	8-битовый элемент данных
DST.	Имя поля назначения данных
DDD.	Содержимое поля назначения данных
SRC.	Имя источника данных
SSS.	Содержимое поля источника данных
RP.	Содержимое регистровой пары
M.	Элемент памяти, поименованный как источник или как место назначения данных

При необходимости для пояснения смысла команд приведенные сокращения будут снабжаться комментариями.

Команды LDA и STA

Двоичное представление

	LDA	STA
Первый байт	00111010	00110010
Второй байт	LSPA	LSPA
Третий байт	MSPA	MSPA

Формат на языке ассемблера

(LABEL:) LDA ADDR

(LABEL:) STA ADDR

Регистр флажков. Не меняется.

В дальнейшем, когда будет рассмотрено больше команд, работающих с данными, будет видно, что для большинства операций один операнд обычно размещается в регистре **A**. Поэтому важно, чтобы данные пересылались из регистра **A** и в него достаточно эффективным образом. Команды загрузки (**LDA**) и пересылки (**STA**) для регистра **A** осуществляют передачу данных между регистром **A** и памятью при условии, что адрес позиции памяти, занимаемой элементом данных (прямая адресация), явно задан. Команда **LDA** используется для выборки данных из памяти и помещения их в регистр, откуда они могут быть взяты для обработки другими командами. После выполнения команды **LDA** значение элемента данных в памяти не меняется. Команда **STA** используется для пересылки результата вычислений из регистра **A** в определенное поле памяти. Одним из простейших применений этих команд является пересылка данных из одного поля памяти в другое. Например:

START: LDA LOG
STA CON

Команда **LDA** осуществляет пересылку данных из поля памяти с именем **LOG** в регистр **A**. Команда **STA** в свою очередь выбирает данные из регистра **A** и направляет их в поле памяти с именем **CON**. Первоначальное содержимое **CON** теряется. Содержимое поля памяти с именем **LOG** остается неизменным.

После выполнения этого фрагмента программы, данные из поля памяти **LOG** передаются в поле памяти **CON**.

Команды LDAX и STAX

Двоичное представление

	LDAX	STAX
Первый байт	00RP1010	00RP0010

RP=00 для регистровой пары В и RP=01 для регистровой пары D.

Формат на языке ассемблера

```
(LABEL:) LDAX B
(LABEL:) LDAX D
(LABEL:) STAX B
(LABEL:) STAX D
```

Регистр флажков. Не меняется.

Команды загрузки аккумулятора (**LDAX**) и пересылки содержимого аккумулятора (**STAX**) похожи на команды **LDA** и **STA**. Отличие состоит в том, что команды **LDAX** и **STAX** реализуют способ адресации по паре регистров, а **LDA** и **STA** — способ прямой адресации. Содержимое либо регистровой пары **B**, либо регистровой пары **D** может быть использовано в качестве адреса для элемента данных. Команды **LDAX** и **STAX** часто используются для выборки данных из последовательного списка, размещенного в памяти, и для пересылки данных из одного списка данных в другой.

```
LDAX B
STAX D
```

Если регистровая пара **B, C** содержит число **0200**, а регистровая пара **D, E** — **0300**, то приведенная выше короткая программа будет пересылать содержимое поля памяти с адресом **0200** в поле памяти с адресом **0300**. Позднее будут рассмотрены команды, позволяющие увеличивать и уменьшать на 1 содержимое регистровых пар **B** и **D**. Это даст возможность последовательно изменять адреса **B** и **D** в соответствии с позициями элементов списка в памяти.

Команда MOV*Двоичное представление*

MOV	
Первый байт	01DDD SSS
DDD или SSS принимают значения: 000—для В, 001—для С, 010—для D, 011—для Е, 100—для Н, 101—для L, 111—для А и 110—для М.	

*Формат на языке ассемблера***(LABEL:) MOV DST, SRC**

где **DST** и **SRC** не могут быть оба одновременно **М**.

Регистр флажков. Не меняется.

Команда **MOV** осуществляет пересылку содержимого одного регистра в другой, т. е. команда **MOV** позволяет использовать для хранения промежуточных данных различные регистры. Так как команда **MOV** занимает в три раза меньше памяти, чем **LDA** и **STA**, и выполняется в три раза быстрее, то часто лучше использовать для временного хранения данных не память, а регистры. Команд, аналогичных **LDA** и **STA**, для других регистров нет. Команда **MOV** вместе с командами **LDA** и **STA** позволяет загружать любой регистр или записывать содержимое любого регистра в память, используя прямую адресацию, за два шага.

Выделим случай, когда в команде **MOV** появляется код **110** либо в позиции регистра-источника, либо в позиции регистра-назначения (сразу в обеих позициях код **110** появиться не может). Надо отметить, что регистра, обозначаемого номером **110**, не существует. Однако когда микропроцессор обнаруживает код **110** в регистровом поле, то считается, что искомый операнд содержится в поле памяти, адрес которого находится в регистровой паре **Н**. Этот метод используется командами разных типов для того, чтобы проводить адресацию по паре регистров **Н** и **L**.

MOV В, М

Эта команда пересылает элемент данных из поля памяти, адрес которого определяется содержимым регистров **Н** и **L**, в регистр **В**.

Команда MVI*Двоичное представление*

MVI	
<hr/>	
Первый байт	00DDD110
Второй байт	DATA, 8

DDD принимает значения: 000—для В, 001—для С, 010—для D, 011—для E, 100—для H, 101—для L, 111—для A и 110—для M.

*Формат на языке ассемблера***(LABEL:) MVI DST, DATA***Регистр флажков.* Не меняется.

Команда пересылки данных (**MVI**) служит для передачи одного байта данных в регистр. Элемент данных — это второй байт в команде **MVI**. Эта команда часто используется для работы с константами. Так же как и в команде **MOV**, в команде **MVI** допускается использование регистровой пары **H** для адресации по паре регистров.

MVI M, 00H

Эта команда служит для передачи байта, состоящего из одних нулей, в поле памяти, адрес которого находится в регистровой паре **H**.

Команда LXI*Двоичное представление*

LXI	
<hr/>	
Первый байт	00RP0001
Второй байт	LSPD
Третий байт	MSPD

RP принимает значения: 00—для В, 01—для D, 10—для H и 11—для SP.

*Формат на языке ассемблера***(LABEL:) LXI B, DATA****(LABEL:) LXI D, DATA****(LABEL:) LXI H, DATA****(LABEL:) LXI SP, DATA**

Регистр флажков. Не меняется.

Команда непосредственной загрузки регистровой пары (LXI) помещает 16-битовый элемент данных в одну из следующих регистровых пар (B, D, H или SP). 16-битовый элемент данных находится во втором и третьем байтах команды LXI. Команда LXI обычно используется для замены адресов в регистровых парах при проведении адресации либо с помощью регистровой пары, либо с помощью стека. Команда LXI может быть также применена для помещения одной 16-битовой или двух 8-битовых констант в регистры.

LXI B, 1122H

Эта команда заносит шестнадцатеричную константу 11 в регистр B и шестнадцатеричную константу 22 в регистр C.

LXI H, TEMP1

MOV M, B

С помощью этого фрагмента программы производится запись в регистр B данных из поля памяти, адрес которого TEMP1. Сначала адрес TEMP1 заносится в регистровую пару H, затем в регистр B передается элемент данных из поля памяти TEMP1. Для этого используется адресация с помощью пары регистров.

Команды LHLD и SHLD

Двоичное представление

	LHLD	SHLD
Первый байт	00101010	00100010
Второй байт	LSPA	LSPA
Третий байт	MSPA	MSPA

Формат языка ассемблера

(LABEL:) LHLD ADDR

(LABEL:) SHLD ADDR

Регистр флажков. Не меняется.

Команды непосредственной загрузки регистров H и L (LHLD) и непосредственной записи в память содержимого регистров H и L (SHLD) похожи на команды LDA и STA. Эти команды осуществляют пересылку данных из регистровой пары H в память, и наоборот. Адрес поля памяти, где находятся младшие 8 бит 16-битового элемента данных, содержится во

втором и третьем байтах команд **LHLD** и **SHLD**. Старшие 8 бит элемента данных находится в поле памяти, адрес которого на 1 больше адреса поля, в котором находятся младшие 8 бит. Сначала из памяти в регистр **L** помещаются младшие 8 бит, а затем в регистр **H** пересылаются старшие биты. Эти команды могут использоваться для временного хранения адреса, ранее содержащегося в паре регистров, в оперативной памяти. Обычно команды **LHLD** и **SHLD** используются для пересылки 16-битовых операндов. В микропроцессоре 8085 фирмы Intel предусмотрено несколько команд, применяющих регистровую пару **H** для операций с 16-битовыми операндами. 16-битовые операции могут использоваться не только для работы с конкретными данными, но и для операций над адресами.

SHLD SAVE

LHLD NEW

Этот фрагмент программы позволяет сохранить содержимое регистров **H** и **L** в двух следующих друг за другом позициях памяти, начиная с адреса **SAVE**. После этого в регистры **H** и **L** пересылаются данные из двух полей памяти с адресом **NEW**.

Команды PUSH и POP

Двоичное представление

	PUSH	POP
Первый байт	11RP0101	11RP000

RP принимает значения: 00 — для B, 01 — для D, 10 — для H, 11 — для PSW.

Формат на языке ассемблера

```
(LABEL:) PUSH B
(LABEL:) PUSH D
(LABEL:) PUSH H
(LABEL:) PUSH PSW
(LABEL:) POP B
(LABEL:) POP H
(LABEL:) POP PSW
```

Регистр флажков. Не меняется.

Команда **PUSH** служит для занесения данных в стек, а команда **POP** — для выталкивания данных из стека. В резуль-

тате выполнения команды **PUSH** в стек добавляется 16-битовый элемент данных. В результате выполнения команды **POP** из стека выталкивается 16-битовый элемент данных. В командах **PUSH** и **POP** присутствует имя пары регистров, которые служат либо источником, либо местом назначения данных. Регистровая пара, служащая указателем стека, не может выступать в роли операнда. Восемь старших разрядов 16-битового операнда помещаются в стек первыми, а выталкиваются последними.

Так, команда **PUSH B** сначала выбирает данные из регистра **B** и помещает их в стек, а затем помещает в стек данные из регистра **C**. Отметим, что из регистра **C** данные помещаются в поле памяти, адрес которого на 1 меньше того адреса, куда попадают данные из регистра **B** (т. е. стек в памяти строится как бы сверху вниз). По команде **POP D** сначала байт данных выталкивается из позиции стека с наименьшим адресом и помещается в регистр **E**, затем выталкиваются данные из следующей позиции стека и помещаются в регистр **D**. Сокращение **PSW** обозначает слово состояния программы. Если в качестве регистровой пары используется **PSW**, то данные берутся из аккумулятора и регистра флажков.

```
PUSH H
PUSH PSW
:
POP PSW
POP H
```

В соответствии с приведенным выше фрагментом программы сначала данные из регистров **H**, **L**, аккумулятора и регистра флажков последовательно помещаются в стек. В дальнейшем в данной программе эти регистры могут использоваться для других целей. Далее восстанавливаются первоначальные значения регистра флажков, аккумулятора, регистров **H** и **L**. Отметим, что данные должны выталкиваться из стека в порядке, обратном их поступлению, т. е. элемент данных, помещенный в стек последним, выталкивается первым.

Перед использованием стека необходимо инициализировать регистр **SP**, указав первый (наибольший) адрес стека. Для этих целей чаще всего используется команда **LXI SP, ADDR**.

Регистр флажков. Ни одна из команд пересылки данных не может воздействовать на состояния флажков. Часто требуется сохранять флажки, сформированные в результате последней операции, даже во время выполнения команд записи результатов и формирования новых данных для следующих операций,

25.9. КОМАНДЫ ОБРАБОТКИ БУЛЕВЫХ ДАННЫХ

Команды обработки булевых данных наиболее часто используются для проведения операций с данными, в которых каждый бит является самостоятельной булевой переменной. Эти команды выполняют операции с данными побитно, т. е. каждый бит одного операнда взаимодействует с битом, занимающим то же самое положение в другом операнде, и результат этой операции присваивается биту, занимающему то же самое положение в результате. Однако операции производятся одновременно со всеми 8 битами каждого байта, который находится либо в регистре, либо в оперативной памяти.

Булевы команды изменяют состояние регистра флажков. Флажки переноса и дополнительного переноса устанавливаются в 0, в то время как флажки знака, нуля и четности принимают свои значения обычным образом. Это значит, что флажок знака устанавливается в 1, если в старшем бите результата булевой операции стоит 1. Флажок нуля устанавливается в 1, если результатом является 0. Флажок четности устанавливается в 1, если результат содержит четное число единиц. Один операнд булевой команды всегда размещается в аккумуляторе. Другой операнд может располагаться в регистре, или в поле памяти, адрес которого определяется регистровой парой или может быть представлен непосредственными данными. Результат всегда помещается в аккумулятор.

Команды ANA и ANI

Двоичное представление

	ANA	ANI
Первый байт	10100SSS	11100110
Второй байт	Нет	DATA 8

SSS принимает значения: 000 — для В, 001 — для С, 010 — для D, 011 — для E, 100 — для H, 101 — для L, 111 — для A и 110 — для M.

Формат на языке ассемблера

(LABEL:) ANA SRC

(LABEL:) ANI DATA

Регистр флажков. Флажки С и АС устанавливаются в 0. Флажки Z, S и P устанавливаются в 1 соответственно, если результат равен нулю, отрицателен или содержит четное число единиц. В противном случае эти флажки сбрасываются в нуль.

Команда **ANA** выполняет операцию логическое И над содержимым аккумулятора и вторым операндом. Результат помещается в аккумулятор. Команда **ANI** также выполняет операцию логическое И, но в этом случае второй операнд находится во втором байте команды.

Каждый бит результата устанавливается в 1, если соответствующие биты операндов равны 1. Функция И может быть использована для выборочной установки в 0 одного или большего числа бит в байте, при этом остальные биты остаются неизменными, т. е. с помощью функции И можно устанавливать в 0 булевы переменные. Для того чтобы установить в 0 переменную, принадлежащую какому-нибудь байту, необходимо выполнить операцию И над этим байтом и маской. Маска содержит 0 в позициях, которые соответствуют положениям тех булевых переменных, которые необходимо установить в 0. Все остальные биты маски равны 1. После выполнения операции И булевы переменные, соответствующие позициям, в которых биты маски равны 0, устанавливаются в 0. Булевы переменные, соответствующие битам маски, равным 1, остаются без изменения.

ANI FEN

С помощью этой команды нулевой бит аккумулятора устанавливается в 0. Остальные биты остаются неизменными.

Функцию И можно использовать также для выбора одного или нескольких битов байта так, чтобы с этими битами можно было работать независимо от других битов этого байта. Например:

MVI A, 0FH

ANA B

Этот фрагмент программы осуществляет выбор четырех первых битов (с 0 по 3) элемента данных, размещенного в регистре **B**, и передает эти биты в соответствующие позиции аккумулятора. Остальные биты аккумулятора (с 4 по 7) будут установлены в 0 независимо от значения битов регистра **B**.

Команды ORA и ORI

Двоичное представление

	ORA	ORI
Первый байт	10110SSS	11110110
Второй байт	Нет	DATA 8

SSS принимает значения: 000 — для В, 001 — для С, 010 — для D, 011 — для Е, 100 — для H, 101 — для L, 111 — для А, 110 — для М.

Формат на языке ассемблера

(LABEL:) ORA SRC
(LABEL:) ORI DATA

Регистр флажков. Флажки **C**, **AC** устанавливаются в 0. Флажки **Z**, **S** и **P** устанавливаются в 1, если результат равен 0, отрицателен или содержит четное число 1 соответственно.

Функция **ИЛИ** — это функция двух логических переменных. Она равна 1, если хотя бы одна булева переменная равна 1. Другими словами, функция **ИЛИ** принимает значение, равное 0, только в том случае, когда оба операнда равны 0. В то время как с помощью функции **И** можно выборочно устанавливать отдельные биты в 0, с помощью функции **ИЛИ** можно выборочно устанавливать отдельные биты в 1. Для этого необходимо провести операцию **ИЛИ** над байтом, содержащим булеву переменную, которую необходимо установить в 1, и над маской. Маска содержит 1 в позиции, соответствующей булевой переменной, которую надо установить в 1. Все остальные биты маски равны 0.

ORI 81H

Эта команда дает возможность установить нулевой и седьмой биты аккумулятора в 1. Остальные биты аккумулятора остаются без изменения.

Команды XRA и XRI*Двоичное представление*

	XRA	XRI
Первый байт	10101SSS	11101110
Второй байт	Нет	DATA 8

SSS принимает значения: 000 — для В, 001 — для С, 010 — для D, 011 — для E, 100 — для H, 101 — для L, 111 — для A и 110 — для M.

Формат на языке ассемблера

(LABEL:) XRA SRC
(LABEL:) XRI DATA

Регистр флажков. Флажки **C**, **AC** устанавливаются в 0. Флажки **Z**, **S** и **P** устанавливаются в 1, если результат равен 0, отрицателен или содержит четное число единиц.

Функция ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR) равна 1, если биты операндов имеют различные значения, она равна 0, если биты операндов одинаковы. Функция ИСКЛЮЧАЮЩЕЕ ИЛИ может быть использована для выполнения операции отрицания над булевыми переменными, т. е. для изменения значения булевой переменной на противоположное: 1 на 0, 0 на 1. Для этого необходимо выполнить операцию XOR над байтом, содержащим искомую переменную, и маской. Маска содержит 1 в позициях, соответствующих переменным, которые необходимо инвертировать. Все остальные биты маски равны 0.

XRI 25H

Эта команда позволяет инвертировать булевы переменные, соответствующие 0, 2 и 5 битам аккумулятора. Остальные биты аккумулятора не меняются.

Команда CMA

Двоичное представление

CMA	
Первый байт	00101111

Формат на языке ассемблера

(LABEL:) CMA

Регистр флажков. Не меняется.

Команда дополнение содержимого аккумулятора (CMA) производит операцию отрицания над содержимым всех битов аккумулятора одновременно. Эта команда не изменяет значения флажков регистра флажков. Она может быть использована для получения дополнительного кода отрицательных чисел (кроме отрицания для получения дополнительного кода необходимо еще добавить 1) и для инвертирования всех битов в байты. Целиком весь байт часто инвертируется по следующим причинам. Это бывает необходимо, во-первых, для подготовки данных к одной из булевых операций, описанных выше, и, во-вторых, для передачи или приема данных от периферийного устройства, которое по каким-либо причинам использует инвертированные данные.

25.10. КОМАНДЫ ЦИКЛИЧЕСКОГО СДВИГА

Команды циклического сдвига используются для изменения позиций битов в байте, обычно для выравнивания булевых переменных в разных байтах при подготовке к проведению булевой

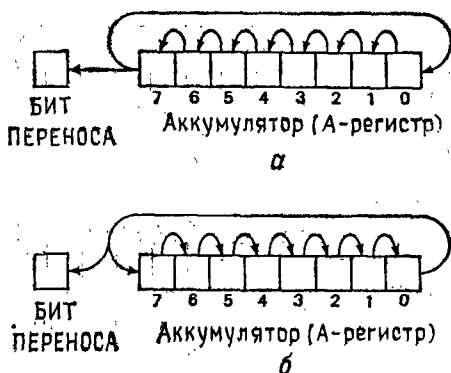


Рис. 25.23. Схемы выполнения команд.

а — команды RLC; б — команды RRC.

операции. Кроме того, сдвиг влево на один разряд эквивалентен умножению на 2, а сдвиг вправо — делению на 2. Правда, только тогда, когда сдвигаемый байт соответствует числу без знака.

Команды циклического сдвига воздействуют только на бит переноса регистра флажков.

Команды RLC и RRC

Двоичное представление

	RLC	RRC
Первый байт	00000111	00001111

Формат на языке ассемблера

(LABEL:) RLC

(LABEL:) RRC

Регистр флажков. Флажки AC, Z, S, P не меняются. Флажок C изменяется так, как описано ниже.

Команды циклического сдвига влево с переносом (RLC) и циклического сдвига вправо с переносом (RRC) проводят сдвиг содержимого аккумулятора влево и вправо соответственно. Бит, который выталкивается из аккумулятора с одного конца, вталкивается в аккумулятор с противоположного конца, чтобы принять участие в завершении операции. Бит, который в данный момент выталкивается из аккумулятора, помещается

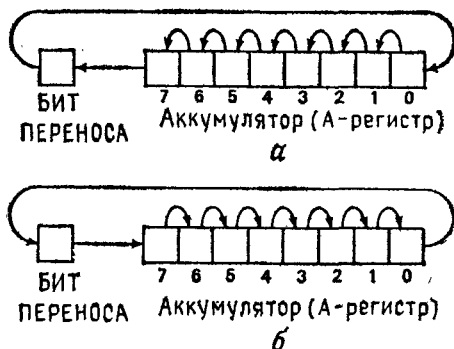


Рис. 25.24. Схемы выполнения команд.
 а — команды RAL; б — команды RAR.

также в регистр флажков на место бита переноса. Отметим, что при выполнении этих команд ни одного бита не теряется и операнд может быть приведен в свое первоначальное состояние. В свою очередь флажку переноса можно присвоить значение, равное значению любого бита аккумулятора. Для этого необходимо только выполнить определенное число команд сдвига.

Схемы выполнения команд **RLC** и **RRC** приведены на рис. 25.2, а, б.

Команды RAL и RAR

Двоичное представление

	RAL	RAR
Первый байт	00010111	00011111

Формат на языке ассемблера

(LABEL:) RAL

(LABEL:) RAR

Регистр флажков. Флажки **AC**, **Z**, **S**, **P** не меняются. Флажок **C** изменяется так, как описано ниже.

Команды циклического сдвига содержимого аккумулятора влево (**RAL**) и вправо (**RAR**) во многом похожи на команды **RLC** и **RRC**. Только по-другому определяется бит, пересылаемый в конец аккумулятора, теперь это бит переноса. На рис. 25.24 приведены схемы выполнения команд **RAL** и **RAR**.

Бит, который выталкивается из аккумулятора, становится битом переноса, а бит переноса вталкивается в аккумулятор. Программист может теперь точно задать значение любого бита, установив в 1 или в 0 бит переноса и выполняя несколько команд **RAL** и **RAR**. Бит, выталкиваемый из аккумулятора по команде сдвига влево или вправо, после следующей аналогичной команды снова помещается в аккумулятор. Такие последовательности команд сдвига можно использовать для сдвига операндов, состоящих из нескольких байтов. Для того чтобы сдвинуть 24-битовый операнд на 1 бит влево, начинать нужно с наименьшего значащего байта. Как это делается описывается ниже.

LDA LSB ;ЗАГРУЗКА МЛАДШЕГО БАЙТА В АККУМУЛЯТОР
RAL ;СЧИТАЕМ, ЧТО БИТ ПЕРЕНОСА РАВЕН НУЛЮ
STA LSB ;СОХРАНЕНИЕ РЕЗУЛЬТАТА
LDA SSB ;ЗАГРУЗКА ВТОРОГО БАЙТА В АККУМУЛЯТОР
RAL ;БИТ, ВЫТОЛКНУТЫЙ РАНЕЕ, ПОПАДАЕТ В ПОСЛЕДНИЙ РАЗРЯД АККУМУЛЯТОРА
STA SSB ;СОХРАНЕНИЕ РЕЗУЛЬТАТА
LDA MSB ;ЗАГРУЗКА СТАРШЕГО БАЙТА В АККУМУЛЯТОР
RAL ;БИТ, ВЫТОЛКНУТЫЙ РАНЕЕ, ПОПАДАЕТ В ПОСЛЕДНИЙ РАЗРЯД АККУМУЛЯТОРА
STA MSB ;СОХРАНЕНИЕ РЕЗУЛЬТАТА

Аналогичная последовательность команд может быть использована для сдвига вправо многобайтовых данных, начиная со старшего разряда.

Команды **STC** и **CMC**

Двоичное представление

	STC	CMC
Первый байт	00110111	00111111

Формат на языке ассемблера

(LABEL:) **STC**
 (LABEL:) **CMC**

Регистр флажков. Флажки **AC**, **Z**, **S** и **P** не меняются. Флажок **C** устанавливается в 1 в случае команды **STC**, и его значение меняется на противоположное при выполнении команды **CMC**. Команды установки в 1 флажка переноса (**STC**) и дополнение содержимого флажка переноса (**CMC**) дают возможность программисту управлять значением этого флажка. Поскольку его значение влияет на выполнение многих операций (например, таких, как **RAL** и **RAR**), возможность управлять им очень важна. Команда **STC** делает значение флажка переноса равным 1. Команда **CMC** меняет значение флажка переноса на противоположное. Хотя команды «очистки» (т. е. установки в 0) флажка переноса не существует, имеется несколько путей, позволяющих провести очистку.

STC

CMC

С помощью этих команд флажок переноса устанавливается в 0.

ORA A

ANA A

Любая из этих двух команд позволяет провести очистку флажков переноса и дополнительного переноса. Содержимое аккумулятора изменяться не будет. Однако значения флажков **Z**, **P** и **S** изменятся, поскольку они отражают состояние аккумулятора.

25.11. КОМАНДЫ ПЕРЕХОДА

Использование команд перехода позволяет изменять обычный последовательный ход выполнения программы. Суть в том, что команды перехода могут изменять значение счетчика команд. Каждая команда перехода занимает три байта памяти. В последних двух байтах хранится новое значение счетчика команд, которое является адресом перехода, т. е. адресом той команды, которую необходимо будет выполнить, когда осуществится переход. Для реализации перехода процессор просто заносит нужный адрес в регистр счетчика команд.

Команда **JMP**

Двоичное представление

JMP	
Первый байт	11000011
Второй байт	LSPA
Третий байт	MSPA

*Формат на языке ассемблера***(LABEL:) JMP ADDR***Регистр флажков. Не меняется.*

Команда **JMP** заставляет микропроцессор продолжать выполнение программы с команды, адрес которой определяется в самой команде **JMP**. Эта команда используется для организации обхода программного сегмента, который не нужно выполнять в данный момент. Команду **JMP** иногда называют командой безусловного перехода, поскольку ее выполнение всегда приводит к переходу на новую ветвь.

Команды JC, JNC, JZ, JNZ, JP, JM, JPE, JPO*Двоичное представление*

	Jxx
Первый байт	11CCCC010
Второй байт	LSPA
Третий байт	MSPA
CCC принимает значения: 000—для JNZ, 001—для JZ, 010—для JNC, 011—для JC, 100—для JPO, 101—для JPE, 110—для JP и 111—для JM.	

Формат на языке ассемблера

(LABEL:) JNZ ADDR
(LABEL:) JZ ADDR
(LABEL:) JNC ADDR
(LABEL:) JC ADDR
(LABEL:) JPO ADDR
(LABEL:) JPE ADDR
(LABEL:) JP ADDR
(LABEL:) JM ADDR

Регистр флажков. Не меняется.

С помощью восьми выше перечисленных команд осуществляется переход, но только в тех случаях, когда истинны условия, определенные в них. Если же условие перехода не удовлетворяется, то микропроцессор просто выполняет следующую по порядку команду. Проверяемые условия непосредственно связаны с состояниями флажков регистра флажков. Поскольку значения

флажков изменяются при выполнении команд, команды условного перехода могут учитывать результаты выполнения многих операций. Ниже приведены восемь команд условного перехода.

JNZ	Переход, если результат не равен нулю ($Z = 0$)
JZ	Переход, если результат равен нулю ($Z = 1$)
JNC	Переход, если нет переноса ($C = 0$)
JC	Переход, если появляется перенос ($C = 1$)
JPO	Переход, если результат содержит нечетное число 1 ($P = 0$)
JPE	Переход, если результат содержит четное число 1 ($P = 1$)
JP	Переход, если результат больше нуля ($S = 0$)
JM	Переход, если результат меньше нуля ($S = 1$)

Команда PCNL

Двоичное представление

PCNL	
Первый байт	11101001

Формат на языке ассемблера

(LABEL:) PCNL

Регистр флажков. Не меняется.

Команда **PCNL** осуществляет пересылку содержимого регистров **H** и **L** в счетчик команд. Команда **PCNL** может быть названа командой безусловного перехода с адресацией по паре регистров. Адрес перехода содержится в регистрах **H** и **L**. С помощью команды **PCNL** может быть организовано множественное ветвление. Для этого необходимо формировать нужный адрес перехода в регистровой паре **H**, используя арифметические и логические команды. Ниже приводится один из вариантов организации множественного ветвления. Во-первых, образуется таблица безусловных переходов. В этой таблице содержатся адреса программ, к одной из которых в каждом конкретном случае и будет осуществляться переход. Адрес перехода для команды **PCNL** образуется из адреса начала таблицы и номера ветви. Так как команды **JMP** размещаются в памяти друг за другом с шагом, равным 4 байт, необходимо умножить номер ветви на 4. Так как адрес начала таблицы равен **00H**, для формирования адреса перехода необходимо пересылать в регистр **L** увеличенный в 4 раза номер программы.

```

      ORG 3C10H
START: LXI  H, TABL ;УСТАНОВКА АДРЕСА НАЧАЛА
                        ;ТАБЛИЦЫ
      LDA  PGMNO     ;ЗАГРУЗКА НОМЕРА ПРОГРАМ-
                        ;МЫ В РЕГИСТР A
      RLC
      RLC            ;УМНОЖЕНИЕ НА 4
      MOV  L, A      ;ПЕРЕСЫЛКА МОДИФИЦИРОВАННОГО АДРЕСА В РЕГИСТР L
      PCHL          ;ПЕРЕДАЧА АДРЕСА В СЧЕТЧИК
                        ;КОМАНД
      ORG 3D00H
TABL: JMP  PGM0      ;ПЕРЕХОД К ПРОГРАММЕ 0
      0             ;ПРОПУСК ОДНОГО БАЙТА ДЛЯ
                        ;ТОГО, ЧТОБЫ АДРЕСА СОСЕД-
                        ;НИХ КОМАНД ОТЛИЧАЛИСЬ
                        ;НА 4 БАЙТ
      JMP  PGM1      ;ПЕРЕХОД К ПРОГРАММЕ 1
      0             ;ПРОПУСК ОДНОГО БАЙТА
      JMP  PGM2      ;ПЕРЕХОД К ПРОГРАММЕ 2
      0
      JMP  PGM3
      0
      JMP  PGM4
      0
      JMP  PGM5
      0
      JMP  PGM6
      0
      JMP  PGM7

```

Разберем пример, который поможет понять работу приведенной выше программы. Если **PGMNO** содержит число 5, то в результате двух операций циклического сдвига это число будет увеличено в 4 раза. Поэтому в регистр **L** попадет десятичное число 20 или **14H**. По команде **PCHL** будет осуществляться переход к команде **JMP PGM5**, которая расположена в области памяти, начальный адрес которой равен **3D14H**. В табл. 25.1 показано, как изменяется номер программы и как он становится составной частью адреса таблицы.

Таблица 25.1. Множественное ветвление

			0000		0NNN	Содержимое PGMNO
0011	H			L		
		1101	0000		0000	Начальный адрес=TABLE
0011	H			L		
		1101	000N		NN00	Модифицированный адрес перехода

Команды ввода (IN) и вывода (OUT)*Двоичное представление*

	IN	OUT
Первый байт	11011011	11010011
Второй байт	Номер регистра ввода-вывода	Номер регистра ввода-вывода

Формат на языке ассемблера

(LABEL:) IN IORN
(LABEL:) OUT IORN

Регистр флажков. Не меняется.

Данные должны быть представлены в форме, пригодной для ввода с внешних устройств и для вывода на них. Команда **IN** осуществляет передачу данных от периферийного устройства или регистра ввода-вывода в аккумулятор. Команда **OUT** осуществляет передачу данных из аккумулятора к периферийному устройству или регистру ввода-вывода.

В микропроцессоре 8085 фирмы Intel для связи с периферийными устройствами предусмотрено 256 входных и 256 выходных портов. Во втором байте команд **IN** и **OUT** содержится адрес того порта (или регистра) ввода-вывода, с которым производится обмен данными.

Команды увеличения (INR) и уменьшения (DCR) на 1*Двоичное представление*

	INR	DCR
Первый байт	00DDD100	00DDD101

DDD принимает значения: 000—для В, 001—для С, 010—для D, 011—для E, 100—для H, 101—для L, 111—для A, 110—для M.

*Формат на языке ассемблера***(LABEL:) INR DST****(LABEL:) DCR DST**

Регистр флажков. Флажок **C** не меняется. Флажок **AC** устанавливается в 1, если формируется перенос из 3 бита. В противном случае флажок **AC** сбрасывается в 0. Флажки **Z**, **P** и **S** устанавливаются в 1, если результат равен нулю, содержат четное число 1 и отрицательны соответственно. В противном случае флажки сбрасываются в 0.

Команды увеличения (**INR**) и уменьшения (**DCR**) содержимого регистра служат для добавления или вычитания 1 из регистра. Кроме того, если используется код **110**, производится операция над содержимым поля памяти, адрес которого определяется с помощью пары регистров. Команды **INR** и **DCR** действуют на все биты регистра флажков, за исключением бита переноса. Кроме того, команды **INR** и **DCR** часто используются для модификации содержимого счетчиков во время организации циклов. Счетчики нужны для управления числом итераций в программных циклах (об этом уже говорилось ранее). Например, многократное прохождение одного и того же участка программы может применяться для реализации временных задержек. Каждый проход занимает точно определенное время. Несколько проходов по циклу позволяют генерировать более длительные временные интервалы.

```

      .
      .
      .
      MVI B, 100D
LOOP: ...
      .
      .
      DCR B
      JNZ LOOP

```

Выше приведенный программный цикл повторяется 100 раз. Сначала в регистр **B** заносится число 100 (цифра **D** говорит о том, что 100 — десятичное число). После очередного выполнения команд, следующих за **LOOP**, содержимое регистра **B** уменьшается на 1. Цикл будет повторяться до тех пор, пока содержимое регистра **B** не станет равным 0. Это произойдет после сотой итерации, и управление перейдет к команде, следующей за **JNZ**.

Команды INX и DCX*Двоичное представление*

	INX	DCX
Первый байт	00RP0011	00RP1011

RP принимает значения: 00—для В, 01—для D, 10—для H и 11—для SP (УС).

Формат на языке ассемблера

```

(LABEL:) INX  B
(LABEL:) INX  D
(LABEL:) INX  H
(LABEL:) INX  SP
(LABEL:) DCX  B
(LABEL:) DCX  D
(LABEL:) DCX  H
(LABEL:) DCX  SP

```

Регистр флажков. Не меняется.

С помощью команд (INX) и (DCX) производится увеличение и уменьшение содержимого регистровой пары. Эти команды используются для изменения адресов, содержащихся в регистровых парах. Поскольку эти команды служат для модификации адресов, они не влияют на значения флажков регистра флажков. Обычно желательно сохранять состояние регистра флажков, сложившееся в результате последней операции над данными, даже во время выполнения команд модификации адресов, необходимых для подготовки новых данных. Ведь результат последней операции над данными часто бывает нужен для проведения очередной операции. Например, при проведении операций сдвига элемента данных, занимающего несколько байтов, необходимо сохранять бит переноса. Программа, изображенная ниже, производит просмотр таблицы, расположенной в памяти, для того чтобы найти строки, состоящие из одних 0. После окончания выполнения программы в регистр D помещается число, равное количеству ненулевых строк таблицы.

```

LXI  D, 000H ;УСТАНОВКА ПАРЫ D, E В 0
LXI  H, TVL  ;ЗАГРУЗКА НАЧАЛЬНОГО АДРЕ-
              ;СА ТАБЛИЦЫ

```

```

LOOP:  MOV  A, M      ;ВЫБОРКА ОЧЕРЕДНОЙ СТРОКИ
                                ;ТАБЛИЦЫ
        ORA  A        ;УСТАНОВКА ФЛАЖКА Z В 1,
                                ;ЕСЛИ A РАВНО НУЛЮ
        JZ   DONE     ;ПЕРЕХОД К DONE, ЕСЛИ СТРОКА
                                ;СОСТОИТ ИЗ НУЛЕЙ
        INX  D        ;ПРИБАВЛЕНИЕ 1 К СЧЕТЧИКУ
                                ;НЕНУЛЕВЫХ СТРОК
        INX  H        ;ПЕРЕХОД К СЛЕДУЮЩЕЙ СТРО-
                                ;КЕ ТАБЛИЦЫ
        JMP  LOOP     ;ПЕРЕХОД К LOOP

DONE:  ...

```

25.12. ПРИМЕР ЛОГИЧЕСКОГО УСТРОЙСТВА УПРАВЛЕНИЯ

На промышленных предприятиях часто возникают задачи управления двигателями и другими устройствами с помощью переключателей, сенсорных клавишей и т. п. Управляющие выходы в этом случае обычно являются булевыми функциями входов. Устройства управления в задачах такого типа могут быть реализованы с помощью средств релейной логики, полупроводниковых элементов или с помощью микроЭВМ, снабженных соответствующими программами. Управляющие функции обычно бывают очень простыми. Например, двигатель должен быть включен, когда включены выключатель и нагревательный прибор.

Можно подумать, что применять микроЭВМ для реализации таких простых функций все равно, что «стрелять из пушки по воробьям». Но попробуем изложить аргументы, говорящие в пользу использования именно микроЭВМ для решения таких задач.

1. МикроЭВМ более надежна, чем схемы релейной логики.
2. В случае изменения функций можно легко и быстро изменить программу микроЭВМ.
3. Хотя каждая функция сама по себе очень проста, для управления насосами, вентиляторами и прочими устройствами потребуется реализовать сотни таких функций. А одна микроЭВМ может заменить сотни релейных схем.
4. Систему, построенную на основе микроЭВМ, легко дополнить и другими возможностями, используя временные задержки, счетчики и т. п.

Предположим, что программа для микроЭВМ должна выполнять функции простого логического устройства управления (контроллера). Она должна вычислять четыре булевых функ-

ции, зависящие от четырех входных переменных. **IA**, **IB**, **IC** и **ID** — имена четырех входных переменных, а **OA**, **OB**, **OC** и **OD** — имена четырех выходных переменных. Ниже приведены формулы, описывающие зависимости выходных переменных от входных.

$$OA = (IA) \cdot (IB) \cdot (IC) \cdot (ID)$$

$$OB = IA + IB + IC + ID$$

$$OC = (IA \cdot IB) + (IC \cdot ID)$$

$$OD = (IA + IB) \cdot (IC \cdot ID)$$

Для размещения значений внешних сигналов будет достаточно двух регистров (ввода-вывода) — одного входного и одного выходного. Ниже показано, как входные и выходные сигналы связаны с портами ввода-вывода.

Функция	Порт ввода-вывода	Биты							
		7	6	5	4	3	2	1	0
ВВОД	02H	0	0	0	0	ID	IC	IB	IA
ВЫВОД	F2H	0	0	0	0	OD	OC	OB	OA

Первым шагом в процессе разработки программы, реализующей функции управления, является построение ее структурной схемы (рис. 25.25). В дальнейшем необходимо до такой степени детализировать действия, относящиеся к каждому блоку схемы, чтобы можно было легко представить их в виде последовательности команд. Для некоторых блоков это сделать очень просто. Например, блок **ВВОД ДАННЫХ** после детализации примет вид, показанный на рис. 25.26. От структурной схемы, изображенной на рис. 25.26, можно непосредственно перейти к фрагменту программы на языке ассемблера.

```
DIN: IN    02H    ;СЧИТЫВАНИЕ ДАННЫХ ИЗ ВХОДНОГО
                ;ПОРТА
        STA INPUT ;ПОМЕЩЕНИЕ ВВЕДЕННЫХ ДАННЫХ
                ;В ПОЛЕ С АДРЕСОМ INPRТ
```

Полный текст программы помещен в конце этого раздела. В процессе обсуждения к нему будут делаться ссылки.

На рис. 25.27 и 25.28 приведены схемы программ, вычисляющих значения **OA** и **OB** соответственно. При выполнении операций **И** и **ИЛИ** для выделения булевых переменных используются команды циклического сдвига. Отметим, что необходимо заранее разработать формат для хранения значений выходных функций. Будем считать, что значения выходных булевых

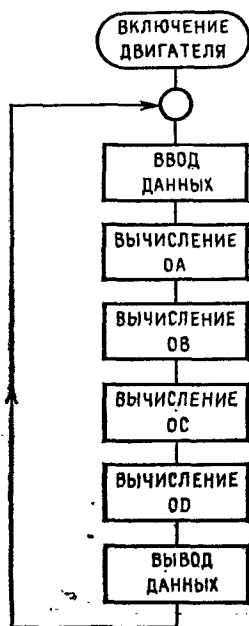


Рис. 25.25. Схема программы логического устройства управления (контроллера).

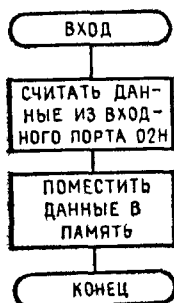


Рис. 25.26. Детализированная схема блока ВВОД ДАННЫХ (рис. 25.25).

функций должны находиться в нулевом бите отведенного для них поля памяти. Все остальные биты этого поля должны быть сброшены в 0.

На рис. 25.29 и 25.30 приведены схемы программ, вычисляющих значения переменных **ОС** и **ОД**, соответственно. Эти программы несколько сложнее предыдущих и в них предусмотрено использование регистров для временного размещения промежуточных значений. Например, программа на рис. 25.29 во время выполнения **1В·1А** временно сохраняет в регистре **Е** значение функции **1С·1D**. Программы, используемые для вычисления значений функций **ОА**, **ОВ**, **ОС** и **ОД** размещаются в памяти, начиная с адресов **РОА**, **РОВ**, **РОС** и **РОД** соответственно.

На рис. 25.31 показана схема программы вывода данных. Цикл здесь используется для того, чтобы все четыре выходных бита заняли правильные позиции в порту вывода. Если разместить значения всех выходных функций в тех же самых позициях и расположить четыре поля памяти, со-

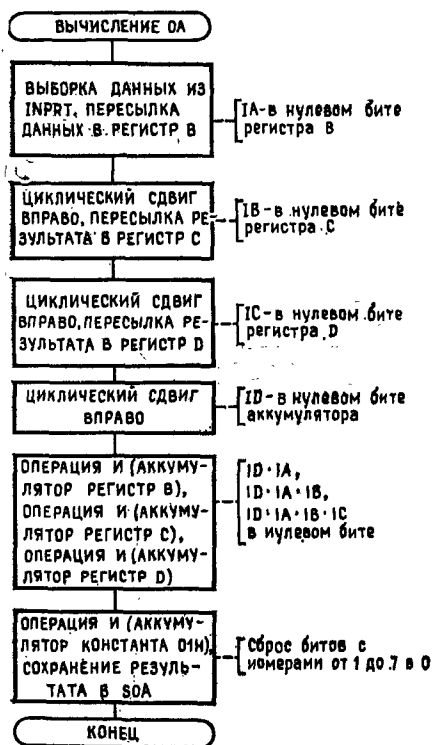


Рис. 25.27. Схема программы, вычисляющей значение функции ОА контроллера.

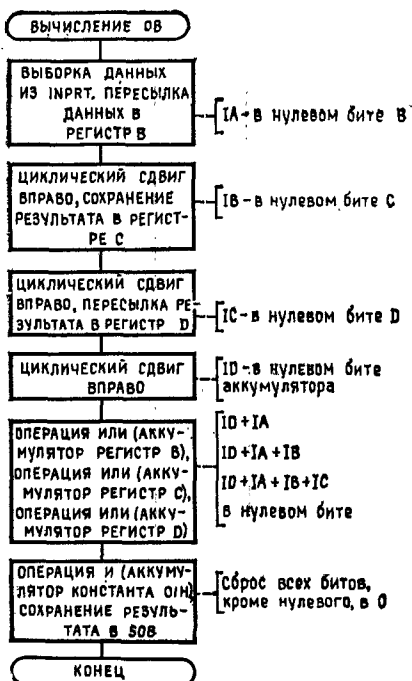


Рис. 25.28. Схема программы, вычисляющей значение функции ОБ контроллера.

державшие эти значения, в определенной последовательности, то с помощью цикла очень легко извлечь из памяти биты со значениями искомым функций и поместить в предназначенные для них позиции. Программа вывода располагается в памяти, начиная с позиции DOUT.

Ниже приведен текст всей программы.

```

DIN:  IN  02H      ;ВЫБОРКА ДАННЫХ ИЗ ВХОДНОГО
                        ;ПОРТА
      STA  INPRT    ;СОХРАНЕНИЕ ДАННЫХ
ROA:  LDA  INPRT    ;ВЫБОРКА ВХОДНЫХ ДАННЫХ
      MOV  BA       ;ПЕРЕСЫЛКА ДАННЫХ В РЕ-
                        ;ГИСТР В
      RRC           ;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                        ;ВЫДЕЛЕНИЕ IB
    
```

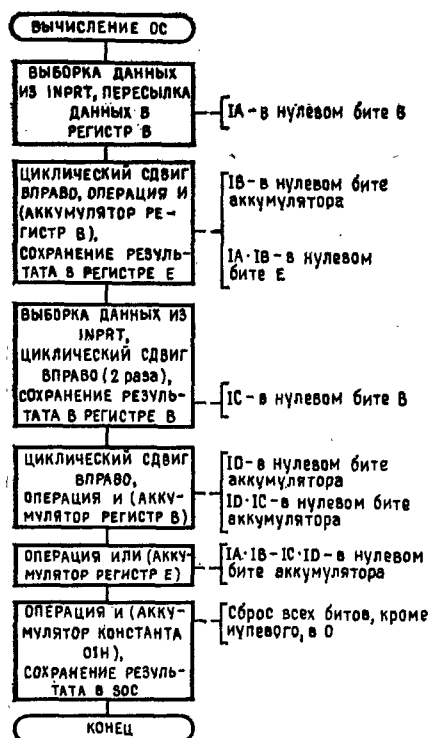


Рис. 25.29. Схема программы, вычисляющей значение функции ОС контроллера.

```

MOV    C, A      ; ПЕРЕСЫЛКА В РЕГИСТР С
RRC                      ; ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                        ; ВЫДЕЛЕНИЕ IC
MOV    D, A      ; ПЕРЕСЫЛКА В РЕГИСТР D
RRC                      ; ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                        ; ВЫДЕЛЕНИЕ ID
ANA    В          ; ОПЕРАЦИЯ И МЕЖДУ ID И IA
ANA    С          ; ОПЕРАЦИЯ И МЕЖДУ ПРЕДЫДУЩИМ РЕЗУЛЬТАТОМ И IB
ANA    D          ; ОПЕРАЦИЯ И МЕЖДУ ПРЕДЫДУЩИМ РЕЗУЛЬТАТОМ И IC
ANI    01Н       ; СБРОС ВСЕХ БИТОВ, КРОМЕ НУЛЕВОГО, В 0
STA    SOА       ; СОХРАНЕНИЕ РЕЗУЛЬТАТА
  
```

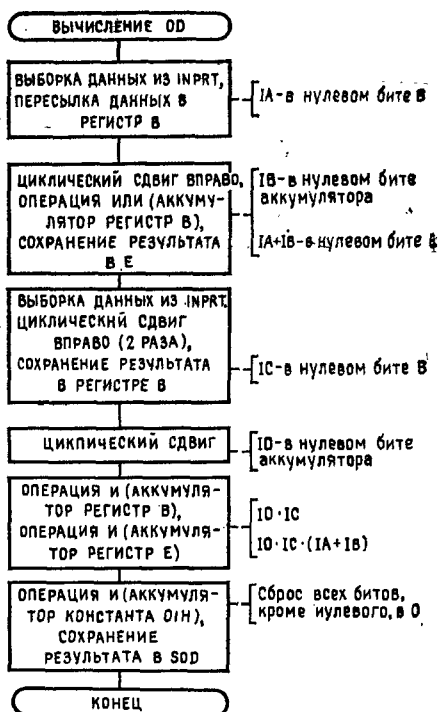


Рис. 25.30. Схема программы, вычисляющей значение функции OD контроллера.

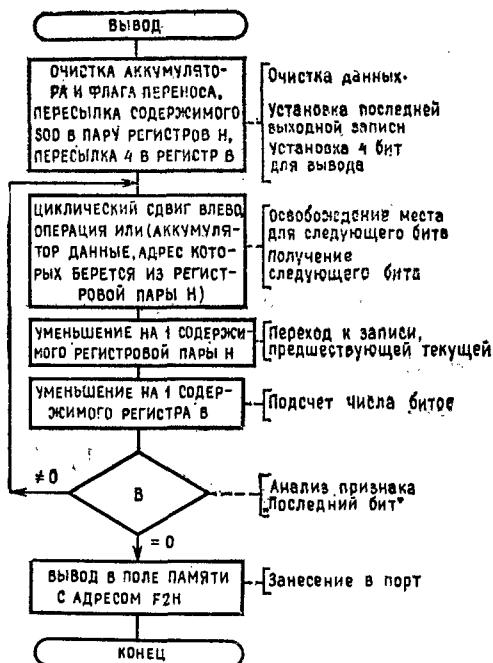


Рис. 25.31. Схема программы вывода контроллера.

```

РОВ: LDA  INPRT ;ВЫБОРКА ВХОДНЫХ ДАННЫХ
      MOV  B, A  ;ПЕРЕСЫЛКА В РЕГИСТР B
      RRC                      ;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                                ;ВЫДЕЛЕНИЕ IB
      MOV  C, A  ;ПЕРЕСЫЛКА В РЕГИСТР C
      RRC                      ;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                                ;ВЫДЕЛЕНИЕ IC
      MOV  D, A  ;ПЕРЕСЫЛКА В РЕГИСТР D
      RRC                      ;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО,
                                ;ВЫДЕЛЕНИЕ ID
      ORA  B      ;ОПЕРАЦИЯ ИЛИ МЕЖДУ ID И IA
      ORA  C      ;ОПЕРАЦИЯ ИЛИ МЕЖДУ ПРЕДЫ-
                                ;ДУЩИМ РЕЗУЛЬТАТОМ И IB
      ORA  D      ;ОПЕРАЦИЯ ИЛИ МЕЖДУ ПРЕДЫ-
                                ;ДУЩИМ РЕЗУЛЬТАТОМ И IC
      ANI  01H    ;СБРОС ВСЕХ БИТОВ, КРОМЕ
                                ;НУЛЕВОГО, В 0
  
```

	STA	SOB	;СОХРАНЕНИЕ РЕЗУЛЬТАТА
POC:	LDA	INPRT	;ВЫБОРКА ВХОДНЫХ ДАННЫХ
	MOV	B, A	;ПЕРЕСЫЛКА В РЕГИСТР В
	RRC		;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО, ;ВЫДЕЛЕНИЕ IB
	ANA	B	;ОПЕРАЦИЯ И МЕЖДУ IA И IB
	MOV	E, A	;СОХРАНЕНИЕ РЕЗУЛЬТАТА В РЕ- ;ГИСТРЕ E
	LDA	INPRT	;ВЫБОРКА ВХОДНЫХ ДАННЫХ
	RRC		;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО
	RRC		;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО, ;ВЫДЕЛЕНИЕ IC
	MOV	B, A	;ПЕРЕСЫЛКА В РЕГИСТР В
	RRC		;ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО, ;ВЫДЕЛЕНИЕ ID
	ANA	B	;ОПЕРАЦИЯ И МЕЖДУ IC И ID
	ORA	E	;ОПЕРАЦИЯ ИЛИ МЕЖДУ ПРЕДЫ- ;ДУЩИМ РЕЗУЛЬТАТОМ И РЕ- ;ГИСТРОМ E
	ANI	01H	;СБРОС ВСЕХ БИТОВ, КРОМЕ НУ- ;ЛЕВОГО, В 0
	STA	SOC	;СОХРАНЕНИЕ РЕЗУЛЬТАТА
POD:	LDA	INPRT	;ВЫБОРКА ВХОДНЫХ ДАННЫХ
	MOV	B, A	;ПЕРЕСЫЛКА В РЕГИСТР В
	RRC		;ВЫДЕЛЕНИЕ IB
	ORA	B	;ОПЕРАЦИЯ ИЛИ МЕЖДУ IA И IB
	MOV	E, A	;ПЕРЕСЫЛКА РЕЗУЛЬТАТА В E
	LDA	INPRT	;ВЫБОРКА ВХОДНЫХ ДАННЫХ
	RRC		;ВЫДЕЛЕНИЕ IC
	MOV	B, A	;ПЕРЕСЫЛКА РЕЗУЛЬТАТА В РЕ- ;ГИСТР В
	RRC		;ВЫДЕЛЕНИЕ ID
	ANA	B	;ОПЕРАЦИЯ И МЕЖДУ ID И IC

ANA	E	;ОПЕРАЦИЯ И МЕЖДУ ПРЕДЫДУЩИМ РЕЗУЛЬТАТОМ И РЕГИСТРОМ E
ANI	01H	;СБРОС ВСЕХ БИТОВ, КРОМЕ НУЛЕВОГО, В 0
STA	SOD	;СОХРАНЕНИЕ РЕЗУЛЬТАТА
DOUT: XRA	A	;ОЧИСТКА АККУМУЛЯТОРА И ФЛАЖКА ПЕРЕНОСА
LXI	H, SOD	;ЗАСЫЛКА В РЕГИСТРОВУЮ ПАРУ АДРЕСА ПОЛЯ, В КОТОРОМ ХРАНИТСЯ ПОСЛЕДНЯЯ ВЫХОДНАЯ ФУНКЦИЯ
MVI	B, 4H	;ЗАСЫЛКА В СЧЕТЧИК ЧИСЛА ИТЕРАЦИИ 4
LOOP: RLC		;ОСВОБОЖДЕНИЕ МЕСТА ДЛЯ СЛЕДУЮЩЕГО БИТА
ORA	M	;ВВОД ЗНАЧЕНИЯ СЛЕДУЮЩЕЙ ВЫХОДНОЙ ФУНКЦИИ
DCX	H	;ПЕРЕХОД К ПРЕДЫДУЩЕЙ ЗАПИСИ
DCR	B	;УМЕНЬШЕНИЕ НА 1 ЗНАЧЕНИЯ СЧЕТЧИКА ИТЕРАЦИИ
JNZ	LOOP	;ПЕРЕХОД К LOOP, ЕСЛИ НЕ РАВНО 0 ЗНАЧЕНИЕ СЧЕТЧИКА ИТЕРАЦИИ
OUT	F2H	;ВЫВОД ЗНАЧЕНИЙ 4 ФУНКЦИИ
JMP	DIN	;ПЕРЕХОД К НАЧАЛУ ПРОГРАММЫ
INPRT: 0		;ПАМЯТЬ ДЛЯ ВХОДНЫХ ДАННЫХ
SOA: 0		;ЗНАЧЕНИЕ ФУНКЦИИ 0A В НУЛЕВОМ БИТЕ
SOB: 0		;ЗНАЧЕНИЕ ФУНКЦИИ 0B В НУЛЕВОМ БИТЕ
SOC: 0		;ЗНАЧЕНИЕ ФУНКЦИИ 0C В НУЛЕВОМ БИТЕ
SOD: 0		;ЗНАЧЕНИЕ ФУНКЦИИ 0D В НУЛЕВОМ БИТЕ



Рис. 25.32. Схема программы контроллера с реализацией поиска по таблице.

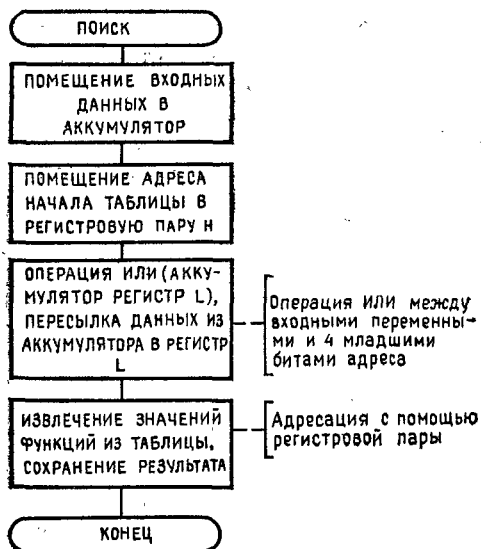


Рис. 25.33. Схема программы поиска данных в таблице.

25.13. ДРУГОЙ ПОДХОД К ПОСТРОЕНИЮ ЛОГИЧЕСКОГО УСТРОЙСТВА УПРАВЛЕНИЯ (КОНТРОЛЛЕРА)

Хотя приведенная в предыдущем параграфе программа правильно выполняет свои функции, ее трудно модифицировать, если изменится логика работы устройства управления. На рис. 25.32 приведена структурная схема программы, которая выполняет те же функции, что и предыдущая, но не вычисляет значения выходных функций, а берет их из таблицы. Поскольку значения функций хранятся в таблице, любые их изменения приведут к изменению лишь содержимого строк таблицы в соответствии с новыми значениями функций. На рис. 25.33 приведена структурная схема программы просмотра таблицы. Начальный адрес таблицы выбирается так, что значения четырех наименьших значащих битов адреса каждой строки таблицы являются значениями входных переменных. Таким образом, каждый из 16 возможных адресов, относящихся к строкам таблицы, соответствует определенной комбинации входных переменных. В каждой строке заранее выделяются четыре бита, значения которых выбираются равными значениям четырех выходных функций для комбинации входных переменных, свя-

занных с этой строкой. Для того чтобы составить таблицу (табл. 25.2), программисту необходимо вычислить все возможные значения четырех функций. Тогда микроЭВМ не нужно будет самой вычислять значения этих функций, а достаточно будет только выбрать нужное из тех значений, которые уже были определены программистом.

Таблица 25.2. Таблица поиска

Адрес	Младшие биты адреса				Функции				Данные в шестнадцатеричной системе
	ID	IC	IB	IA	OD	OC	OB	OA	
3D50	0	0	0	0	0	0	0	0	00
3D51	0	0	0	1	0	0	1	0	02
3D52	0	0	1	0	0	0	1	0	02
3D53	0	0	1	1	0	1	1	0	06
3D54	0	1	0	0	0	0	1	0	02
3D55	0	1	0	1	0	0	1	0	02
3D56	0	1	1	0	0	0	1	0	02
3D57	0	1	1	1	0	1	1	0	06
3D58	1	0	0	0	0	0	1	0	02
3D59	1	0	0	1	0	0	1	0	02
3D5A	1	0	1	0	0	0	1	0	02
3D5B	1	0	1	1	0	1	1	0	06
3D5C	1	1	0	0	0	1	1	0	06
3D5D	1	1	0	1	1	1	1	0	0E
3D5E	1	1	1	0	1	1	1	0	0E
3D5F	1	1	1	1	1	1	1	1	0F

Ниже приведен текст всей программы. Поскольку четырьмя младшими битами адреса таблицы являются 0000, для формирования нужного адреса четыре бита, представляющие значения входных переменных, вступают в операцию ИЛИ с адресом таблицы.

```

      ORG    3C10H
STRT: IN     02H      ;ВВОД ДАННЫХ
      LXI    H, TABL  ;АДРЕС НАЧАЛА ТАБЛИЦЫ
      ORA    L        ;ВСТАВКА ВХОДНЫХ ДАННЫХ
                        ;В ПОЗИЦИИ 4 МЛАДШИХ БИ-
                        ;ТОВ
      MOV    L, A      ;МОДИФИКАЦИЯ АДРЕСА, ХРА-
                        ;НЯЩЕГОСЯ В РЕГИСТРОВОЙ
                        ;ПАРЕ H
      MOV    A, M      ;ПЕРЕСЫЛКА В АККУМУЛЯТОР
                        ;СТРОКИ ТАБЛИЦЫ

```

```

OUT    F2H      ;ВЫВОД ЗНАЧЕНИЙ ФУНКЦИЙ
JMP     STRT     ;ПЕРЕХОД К НАЧАЛУ ПРОГ-
                ;РАММЫ

ORG     3D50H

TABL:   00H      ;ТАБЛИЦА ЗНАЧЕНИЙ ФУНК-
                ;ЦИИ

        02H
        02H
        06H
        02H
        02H

```

Отметим, что преимущество этой программы перед программой, приведенной ранее, не только в том, что значения функций могут быть легко изменены, но и в том, что она короче и проще. Небольшая длина и простота программы способствуют экономии памяти и времени, затрачиваемого на разработку программы, увеличивают скорость ее выполнения и уменьшают вероятность появления ошибок, т. е. программа очень удобна для реализации на микроЭВМ. Очевидно, что использование таблиц для поиска в них данных или для организации множественного ветвления всегда будет обеспечивать упомянутые выше преимущества. Поэтому рекомендуется всегда, когда это возможно, применять таблично-ориентированные (или таблично-управляемые) методы программирования.

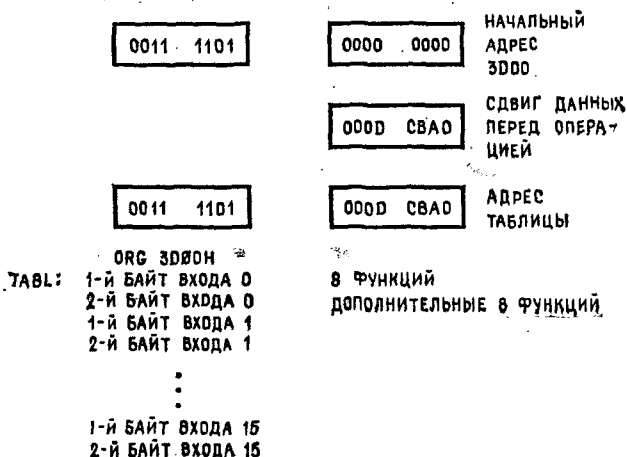


Рис. 25.34. Пример увеличения размера строк таблицы.

В каждой строке таблицы из 8 бит 4 остаются неиспользуемыми. Поэтому легко дополнить систему еще 4 функциями. Таким образом, с помощью этой таблицы могут быть представлены 8 различных функций четырех переменных. Если перед формированием адреса строки таблицы входные данные сдвинуть влево на 1 разряд, то каждая входная комбинация будет определять не однобайтную, как ранее, а двухбайтную строку таблицы. В строке такого размера можно разместить значения 16 булевых функций (рис. 25.34). Но, с другой стороны, если удвоить число выходных переменных, то для размещения 8 функций 8 переменных понадобится уже 256 байт памяти, т. е. размер таблицы увеличивается в 2 раза, когда удваивается число выходов и когда количество входных переменных увеличивается на 1. Поэтому часто при применении таблично-управляемых методов программирования затрачивается много памяти. Заметим, что стоимость единицы памяти быстро падает, а затраты на проектирование систем, которые эту память используют, растут. Поэтому целесообразно экономить средства, затрачиваемые на проектирование за счет дополнительного расхода памяти.

Очень часто размер таблицы удастся уменьшить. Например, этого можно добиться делением большой таблицы на более мелкие. Как уже отмечалось, таблица для восьми функций, зависящих от восьми переменных, занимает 256 байт. Допустим, в результате анализа функций выяснилось, что две функции зависят только от пяти переменных, а остальные шесть — от двух. Тогда для размещения первых двух функций потребуется 32 байт, а шести оставшихся — всего 8 байт. Для двух независимых таблиц потребуется таким образом только 40 байт памяти.

ПОДПРОГРАММЫ, ПРЕРЫВАНИЯ И АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ¹⁾

К. Виатровски, Ч. Хаус

26.1. ПОДПРОГРАММЫ

Подпрограмма — это часть программы, которая может неоднократно использоваться при выполнении алгоритма и занимает в ЭВМ определенный участок памяти. В виде подпрограмм можно оформлять часто используемые последовательности команд, что позволяет экономить память, занимаемую программой. Кроме того, применение подпрограмм дает возможность проводить разработку, программирование и отладку модулей, реализующих типовые функции, независимо друг от друга. Из таких модулей формируется библиотека стандартных подпрограмм, которые затем могут быть включены в самые различные программы.

Подпрограмма выполняется как последовательность шагов программы и вызывается в некоторой точке алгоритма, когда возникает такая необходимость. После завершения подпрограммы происходит **возврат** на тот шаг программы, который следует за вызовом подпрограммы (рис. 26.1). Одна подпрограмма может вызываться многократно из разных точек основной программы. При этом каждый раз в вызываемой подпрограмме должно быть организовано запоминание точки программы, из которой производился ее вызов, чтобы затем правильно выполнить возврат из подпрограммы. Для запоминания адреса возврата из подпрограммы требуется одна 16-разрядная ячейка памяти.

На рис. 26.2 приведен пример, в котором одни подпрограммы вызывают другие. Перед вызовом второй подпрограммы для нее должен быть сформирован адрес возврата. При выходе из второй подпрограммы процессору должен быть указан именно этот адрес возврата. Для организации такой связи между подпрограммами, когда обеспечивается возможность правильного вызова третьей подпрограммы из второй, четвертой —

¹⁾ Adapted from Logic Circuits and Microcomputer Systems, by Claude A. Wiatrowski and Charles H. House. Copyright © 1980. Used by permission of McGraw-Hill, Inc. All rights reserved.

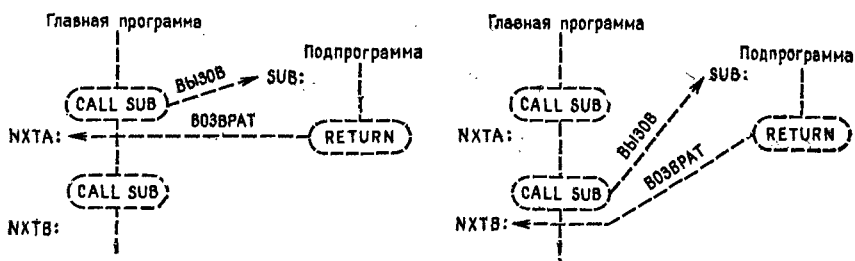


Рис. 26.1. Организация вызовов подпрограммы и возвратов в главную программу.

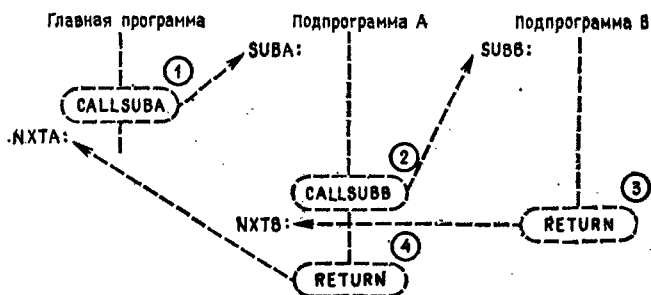


Рис. 26.2. Передача управления вложенным подпрограммам.

из третьей и т. д., необходимо создать соответствующий механизм. В общем случае решение этой задачи просто за счет запоминания последовательности адресов возврата может оказаться неудачным.

Таким образом, в памяти ЭВМ необходимо зарезервировать некоторый участок памяти, куда можно было бы в любое время записывать адрес возврата независимо от того, сколько адресов в него уже записано. При этом обратная выборка адресов должна выполняться в таком порядке: последний адрес возврата должен выбираться первым, предпоследний — вторым и т. д. Ранее был рассмотрен метод построения списка переменной длины, в котором запоминание и выборка данных осуществляется по принципу «последний пришел — первый вышел». Список с такой организацией называется **стеком**. В микропроцессоре серии 8085 фирмы Intel для запоминания адресов возврата из подпрограмм используется именно стековый принцип.

На рис. 26.3 представлен механизм действия стека для примера вложенных подпрограмм на рис. 26.2. При вызове подпрограммы SUBA адрес NXТА следующей за вызовом под-

Команды CALL, CC, CNC, CZ, CNZ, CP, CM, CPE и CPO*Двоичное представление*

	CALL	Cxx
Первый байт	11001101	11CCC100
Второй байт	LSPA	LSPA
Третий байт	MSPA	MSPA

CCC принимает следующие значения: для команды CNZ—000, для CZ—001, для CNC—010, для CC—011, для CPO—100, для CPE—101, для CP—110, для CM—111.

Формат на языке ассемблера

```
(LABEL:) CALL ADDR
(LABEL:) CNZ  ADDR
(LABEL:) CZ   ADDR
(LABEL:) CNC  ADDR
(LABEL:) CC   ADDR
(LABEL:) CPO  ADDR
(LABEL:) CPE  ADDR
(LABEL:) CP   ADDR
(LABEL:) CM   ADDR
```

Регистр флажков. Не изменяется.

При вызове подпрограммы с помощью любой из этих команд сначала в стек записывается содержимое счетчика команд, представляющее собой адрес команды, следующей за командой вызова CALL. Затем в счетчик команд заносится адрес вызываемой подпрограммы, который указан во втором и третьем байтах команды вызова. Для выполнения команд условного вызова подпрограмм дополнительно должны выполняться условия, аналогичные тем, которые используются в командах условного перехода, описанных в предыдущей главе.

Команды RET, RC, RNC, RZ, RNZ, RP, RM, RPE и RPO*Двоичное представление*

	RET	Rxx
Первый байт	11001001	11CCC000

CCC принимает следующие значения: для команды RNZ—000, для RZ—001, для RNC—010, для RC—011, для RPO—100, для RPE—101, для RP—110, для RM—111.

Формат на языке ассемблера

(LABEL:) RET
(LABEL:) RNZ
(LABEL:) RZ
(LABEL:) RNC
(LABEL:) RC
(LABEL:) RPO
(LABEL:) RPE
(LABEL:) RP
(LABEL:) RM

Регистр флажков. Не изменяется.

При выполнении данных команд 16-разрядный адрес возврата пересылается из стека в счетчик команд.

Организация данных для подпрограмм

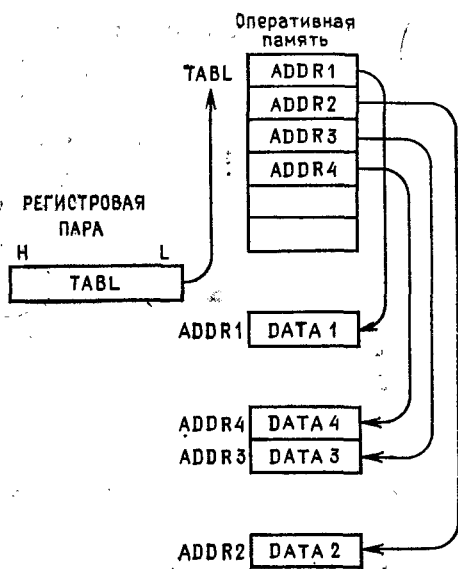
Некоторые подпрограммы каждый раз, когда они вызываются, используют данные, хранящиеся в одних и тех же областях памяти. Как правило, для разных вызовов подпрограмм по крайней мере часть этих данных изменяется. Такие изменяющиеся данные называются **аргументами** подпрограммы, и говорят, что при вызове аргументы передаются в подпрограмму. Передача аргументов в подпрограмму может быть выполнена несколькими способами.

Если в подпрограмму должно быть передано небольшое количество данных, то перед вызовом передаваемые данные можно записывать в регистры, а затем во время выполнения подпрограммы считывать их. Один байт проще передавать в подпрограмму с помощью аккумулятора. Разумеется, обратная передача результата вычислений в главную программу может быть выполнена тем же способом. В случае, если в подпрограмму должны передаваться **массивы** данных, для обеспечения доступа в подпрограмме к элементам массивов достаточно в паре регистров указать начальные адреса этих массивов.

Если данные, передаваемые в подпрограмму, занимают не последовательно расположенные позиции памяти, а рассредоточены по различным ее участкам, то применяется метод передачи данных, схема которого изображена на рис. 26.4. В этом случае в главной программе формируется и размещается в памяти список адресов всех элементов данных. В вызываемую подпрограмму через пару регистров передается адрес начала этого списка адресов. Далее за счет двукратного выполнения команды **MOV** адрес из сформированной таблицы загружается в соответствующую пару регистров, используемых командами

STAX или **LDAX**, с помощью которых выполняется обращение к нужному элементу. Описанный метод представляется достаточно сложным, но в ряде случаев он может оказаться единственным возможным.

Кроме того, любые данные за счет использования регистров могут быть помещены в стек. Тогда перед извлечением этих данных из стека подпрограмма должна выбрать и запомнить адрес возврата, а перед выходом из подпрограммы адрес возврата должен быть помещен обратно в стек.



Использование регистров

Рис. 26.4. Использование списка адресов для передачи данных в подпрограмму.

При составлении программы программист определяет порядок использования в ней регистров. Для вызываемой программы содержимое регистров не должно изменяться, т. е. после возврата из вызываемой подпрограммы содержимое регистров должно оставаться тем же, что и перед ее вызовом, независимо от того, использовались регистры в подпрограмме или нет. В простейшем случае любая подпрограмма использует по крайней мере аккумулятор.

Для того чтобы после возврата из подпрограммы первоначальное содержимое регистров не искажалось, в подпрограмме необходимо предусмотреть сохранение этих регистров перед их использованием. Исходное содержимое регистров восстанавливается перед возвратом в главную программу. Сохранение регистров может быть выполнено различными способами, но наиболее простым представляется запоминание значений регистровых пар в стеке.

Пример составления подпрограммы

В рассматриваемом примере вызов подпрограммы **SUBR** выполняется из двух различных точек главной программы. Вторая по порядку команда вызова **CALL** находится в теле цикла. Внутри цикла только аккумулятор может использоваться в ка-

честве счетчика, поскольку в вызываемой подпрограмме предусматривается сохранение и восстановление аккумулятора.

```

CALL SUBR ;ВЫЗОВ ПОДПРОГРАММЫ
:
:
LOOP: MVI A, 10D
CALL SUBR ;ВЫЗОВ ПОДПРОГРАММЫ ЕЩЕ
;10 РАЗ
DCR A
JNZ LOOP
SUBR: PUSH PSW ;СОХРАНЕНИЕ АККУМУЛЯТОРА
;И РЕГИСТРА ФЛАЖКОВ
:
POP PSW ;ВОССТАНОВЛЕНИЕ АККУМУЛЯ-
;ТОРА И РЕГИСТРА ФЛАЖКОВ
RET ;ВОЗВРАТ В ГЛАВНУЮ ПРОГ-
;РАММУ

```

Все данные, помещенные в стек, должны быть извлечены оттуда перед возвратом в главную программу. Тогда при выполнении команды возврата указатель стека будет определять верный адрес возврата. Подпрограмма, рассмотренная в приведенном примере, сама может вызвать другую подпрограмму. В этом случае соответствующий адрес возврата помещается в стек вслед за содержимым регистра **PSW**. При возврате из второй подпрограммы в первую этот адрес извлекается из стека. После описанных действий указатель стека вновь будет определять положение в стеке регистра **PSW**, что обеспечит непосредственный доступ к содержимому **PSW**.

26.2. ПРЕРЫВАНИЯ

Во время вычислений в процессор поступают внешние сигналы, которые заставляют процессор прекратить обработку и начать выполнять какие-то другие действия. Такие сигналы называются сигналами прерываний. Прерывания необходимы в тех случаях, когда в микроЭВМ от внешних периферийных устройств поступают запросы на выполнение операций, не допускающих ожидания окончания вычислений.

По сигналу прерывания процессору предписывается выполнить некоторую специальную программу, вызов которой производится таким же способом, как в случае обычной подпрограммы. Если приходит сигнал прерывания, то после выполнения те-

кущей команды процессор останавливается, а система прерываний выключается, т. е. все остальные прерывания блокируются. При этом в счетчике команд находится адрес следующей команды, которая должна выполняться после завершения подпрограммы обработки прерывания. Этот адрес помещается в стек, а в счетчик команд заносится адрес подпрограммы обработки прерываний. После выполнения указанных операций процессор начинает обработку возникшего прерывания. Возврат из подпрограммы обработки прерываний в главную программу осуществляется с помощью команды **RET**, по которой адрес возврата извлекается из стека и записывается в счетчик команд.

Каждому типу прерывания поставлен в соответствие начальный адрес подпрограммы обработки этого прерывания. Значения этих адресов приведены в табл. 26.1. Переход в подпрограмму обработки прерываний по соответствующему адресу

Таблица 26.1. Типы прерываний в микропроцессоре 8085 фирмы Intel

Приоритет	Обозначение прерывания	Адрес	Комментарий
Высший	TRAP	24H	Не может быть запрещено
	RST7.5	3CH	Внутренний триггер прерывания реагирует на изменение фронта сигнала
	RST6.5	34H	
	RST5.5	2CH	
Низший	INTR	См. ниже	Используется только в микропроцессоре 8080 фирмы Intel
	RST0	00H	Всем этим прерываниям приспаны одинаковые приоритеты, но соответствующие им программы обработки прерываний имеют разные начальные адреса
	RST1	08H	
	RST2	10H	
	RST3	18H	
	RST4	20H	
	RST5	28H	
	RST6	30H	
	RST7	38H	

осуществляется с помощью команды **JMP**. Пяти типам прерываний присвоены определенные приоритеты, значения которых также приведены в табл. 26.1. Если одновременно поступает несколько сигналов прерывания, то процессор в первую очередь обрабатывает прерывание с наивысшим приоритетом. Затем производится обработка прерывания со следующим по порядку приоритетом и т. д.

Инициализация

Для инициализации микропроцессор 8085 фирмы Intel использует элемент памяти с адресом 0. Именно по этому адресу, после включения питания микропроцессора помещается первая

выполняемая команда. Как правило, этой командой является команда **JMP**, осуществляющая переход на начало главной программы.

Команды EI и DI

Двоичное представление

	EI	DI
Первый байт	11111011	11110011

Формат на языке ассемблера

(LABEL:) EI

(LABEL:) DI

Регистр флажков. Не изменяется.

При поступлении в процессор сигнала прерывания происходит автоматическая блокировка системы прерываний. Это делается по двум причинам. Во-первых, причина прерывания должна быть устранена до запуска программы обработки прерывания (обычно с помощью сброса соответствующего триггера в периферийном устройстве). В противном случае это периферийное устройство будет вызывать прерывание подпрограммы обработки своего же прерывания. Тогда процессор окажется полностью неработоспособным, поскольку постоянно будет реагировать только на сигналы прерываний. Во-вторых, обычно нежелательно, чтобы какое-либо внешнее устройство вызывало прерывание программы, начавшей обработку прерывания, поступившего от другого периферийного устройства. Но, если все-таки требуется, чтобы какое-либо устройство имело возможность прерывать выполнение программы обработки прерывания, поступившего от другого устройства, необходимо предусмотреть специальные средства.

Для того чтобы обеспечить обработку любых дополнительных прерываний во время выполнения программы обработки прерывания, система прерывания должна быть приведена в рабочее состояние. Этой цели служат команда **EI** разрешения прерываний. Данная команда всегда является последней выполняемой командой перед командой возврата **RET**. Система прерываний не блокируется, пока не закончится выполнение команды, следующей за командой разрешения прерываний. За счет выполнения команды **EI** непосредственно перед командой возврата гарантируется, что нового прерывания не возникнет, если адрес возврата еще не извлечен из стека и не записан в счетчик команд, а микропроцессор перейдет к выполнению со-

ответствующей команды основной программы. Таким образом, перед обслуживанием нового прерывания будет выполнена по крайней мере одна команда основной программы. Адрес возврата в основную программу всегда будет иметь верное значение.

Команда **DI** запрещения прерываний дает возможность при необходимости маскировать прерывания в главной программе. Потребность в такой команде объясняется тем, что главная программа может участвовать в вычислительном процессе со строгими временными ограничениями или использоваться в вычислениях, не допускающих прерываний, а иногда нужно просто игнорировать сигналы прерываний от периферийных устройств, если нет необходимости в их обработке. Кроме того, главная программа и программа обработки прерываний могут обмениваться значениями аргументов при передаче их через определенные поля памяти, а также могут совместно использовать более одного аргумента. При этом пока главная программа занята считыванием или изменением соответствующих аргументов, прерывания должны быть запрещены. Пусть, например, прерывания не были запрещены, а главная программа до возникновения прерывания успела изменить только два из трех связанных с ней аргументов. Тогда программа обработки прерываний будет использовать обновленные значения первых двух аргументов и старое значение третьего аргумента.

Следует отметить, что прерывание **TRAP** не блокируется командой **DI**. Прерывание **TRAP** в основном используется в аварийных ситуациях, когда требуется немедленное обслуживание этого прерывания, несмотря на наличие других прерываний. Например, прерывание **TRAP** возникает, если вышел из строя блок питания микроЭВМ. Обычно в источниках питания в таких случаях остается достаточно энергии, чтобы обеспечить функционирование микроЭВМ в течение нескольких миллисекунд. За это время микроЭВМ может заблокировать периферийные устройства и перевести систему в безопасное состояние до полного отключения питания.

Команды **RIM** и **SIM**

Двоичное представление

	RIM	SIM
Первый байт	00100000	00110000

Формат на языке ассемблера

(LABEL:) **RIM**
(LABEL:) **SIM**

Регистр флажков. Не изменяется.

Команда считывания маски прерываний **RIM** используется для передачи в аккумулятор разрядов внутреннего состояния процессора, определяемых в соответствии с табл. 26.2 А с помощью команды установки маски прерываний **SIM** разрядам

Таблица 26.2. Форматы команд **RIM** и **SIM**

Номер разряда	Имя	Формат команды RIM
7	SID	Последовательные входные данные
6	I7.5	Прерывание типа 7.5 разрешено, если разряд равен 1
5	I6.5	Прерывание типа 6.5 разрешено, если разряд равен 1
4	I5.5	Прерывание типа 5.5 разрешено, если разряд равен 1
3	IE	Прерывание всех типов разрешено, если разряд равен 1
2	M7.5	Прерывание типа 7.5 запрещено, если разряд равен 1
1	M6.5	Прерывание типа 6.5 запрещено, если разряд равен 1
0	M5.5	Прерывание типа 5.5 запрещено, если разряд равен 1

Номер разряда	Имя	Формат команды SIM
7	SOD	Последовательные выходные данные
6	SOE	Последовательный вывод разрешен, если разряд равен 1
5	X	
4	R7.5	Сброс триггера прерывания типа 7.5, если разряд равен 1
3	MSE	Установка маски разрешена, если разряд равен 1
2	M7.5	Прерывание типа 7.5 маскировано (запрещено), если разряд равен 1
1	M6.5	Прерывание типа 6.5 маскировано (запрещено), если разряд равен 1
0	M6.5	Прерывание типа 5.5 маскировано (запрещено), если разряд равен 1

состояния процессора присваиваются значения соответствующих разрядов аккумулятора.

Разряды маски прерываний **M7.5**, **M6.5** и **M5.5** могут быть считаны командой **RIM** и изменены посредством команды **SIM**. С помощью маски прерываний можно запретить некоторые типы прерываний, соответствующие определенным разрядам маски. В то время когда обрабатываются прерывания от какого-либо периферийного устройства, заявки на обслуживание со стороны других устройств могут быть пропущены. Изменение разрядов маски с помощью команды **SIM** происходит лишь тогда, когда разряд **MSE** равен 1. Так делается потому, что команда **SIM** используется для трех разных функций, и представляется нерациональным всякий раз, когда выполняется эта команда, вносить изменения во все эти три функции.

Разряд **IE** команды **RIM** показывает, включена или нет система прерываний. С помощью команды **EI** этот разряд устанавливается равным 1, с помощью команды **DI** — равным 0. Команды **EI** и **DI** обеспечивают соответственно включение и отключение всех прерываний типа **RST7.5**, **RST6.5**, **RST5.5** и **INTR**. Команда **SIM** используется для выборочного включения и отключения прерываний **RST7.5**, **RST6.5** и **RST5.5**. Прерывание типа **TRAP**, естественно, не может быть запрещено.

В ряде случаев, когда прерывания запрещены, полезно уметь определять, прерывания какого типа произошли. Это делается с помощью разрядов **I7.5**, **I6.5** и **I5.5** команды **RIM**, каждый из которых соответствует определенному прерыванию. Например, если разряд **I6.5** равен 1, то произошло прерывание типа **RST6.5**.

Вход **RST7.5** чувствителен к фронту сигнала. При появлении переднего фронта логического сигнала на этом входе происходит установка внутреннего триггера прерывания, даже если сигнал прерывания **RST7.5** отсутствует. Этот триггер сбрасывается автоматически по команде **RET** после обработки прерывания **RST7.5**. Сброс указанного триггера может быть выполнен также установкой в 1 разряда **R7.5** команды **SIM**.

Кроме того, команды **RIM** и **SIM** используются для управления одноразрядным входным портом **SID** и одноразрядным выходным портом **SOD**, которые встроены в процессор. Команда **RIM** считывает значение из входного порта в разряд **SID**. По команде **SIM** в выходной порт заносится значение разряда **SOD**. Значение разряда **SOD** изменяется только в том случае, если разряд **SOE** равен 1. За счет этого установка масок прерываний с помощью команды **SIM** может выполняться без воздействия на выходной порт **SOD**. Разряды **SID** и **SOD** называются последовательными разрядами данных. По своей природе ни один из этих разрядов не может быть последовательным, так как каждый из них представляет собой обычный одноразрядный порт. Последовательными они называются потому, что могут быть использованы программой для последовательного ввода и вывода данных.

Использование регистров в случае прерываний

Поскольку прерывание программы может произойти в произвольный момент времени (если, конечно, прерывания не запрещены), то программа обработки прерываний должна обеспечивать сохранение значений **PSW** и аккумулятора в стеке и восстановление их перед возвратом в главную программу. Содержимое всех остальных регистров, как правило, также должно сохраняться и затем восстанавливаться.

Пример программы с прерываниями

В приводимом примере с помощью подпрограммы INTR производится регистрация прерываний типа RST3.

```

      ORG    0H
      JMP    STRT      ;ВКЛЮЧЕНИЕ ПИТАНИЯ
      ORG    18H
      JMP    INTR      ;ВОЗНИКНОВЕНИЕ ПЕРЕРЫВА-
                        ;НИЯ
STRT:  ORG    400H
      MVI    A, 10D    ;ЗАПИСЬ В СЧЕТЧИК ЧИСЛА 10
      STA    COUNT
      LXI    ST, TOP   ;ИНИЦИАЛИЗАЦИЯ УКАЗАТЕЛЯ
                        ;СТЕКА
      EI      ;РАЗРЕШЕНИЕ ПЕРЕРЫВАНИЙ
WAIT:  LDA    COUNT    ;ОЖИДАНИЕ 10 ПЕРЕРЫВАНИЙ
      ORA    A          ;ПРОВЕРКА СЧЕТЧИКА
      JNZ    WAIT
      DI      ;ЗАПРЕЩЕНИЕ ПЕРЕРЫВАНИЙ,
                        ;ЕСЛИ В ОСТАВШЕЙСЯ ЧАСТИ
                        ;ПРОГРАММЫ ОНИ НЕ НУЖНЫ
      ;
INTR:  PUSH   PSW       ;СОХРАНЕНИЕ АККУМУЛЯТОРА
                        ;И РЕГИСТРА ФЛАЖКОВ
      LDA    COUNT     ;ПОДСЧЕТ ЧИСЛА ПЕРЕРЫВА-
                        ;НИЙ
      DCR    A          ;УМЕНЬШЕНИЕ СЧЕТЧИКА НА 1
      STA    COUNT
      POP    PSW        ;ВОССТАНОВЛЕНИЕ АККУМУ-
                        ;ЛЯТОРА И РЕГИСТРА ФЛАЖ-
                        ;КОВ
      EI      ;РАЗРЕШЕНИЕ ПЕРЕРЫВАНИЙ
      RET      ;ВОЗВРАТ

```

Команды HLT и NOP

Двоичное представление

	HLT	NOP
Первый байт	01110110	00000000

Формат на языке ассемблера

(LABEL:) HLT

(LABEL:) NOP

Регистр флажков. Не изменяется.

Команда остановки **HLT**, которую лучше было бы называть командой ожидания прерываний, переводит процессор в состояние останова до тех пор, пока не появится какое-либо прерывание. После возврата из программы обработки прерываний процессор выполняет команду, следующую за командой **HLT**.

Команда **NOP** (нет операции) не выполняет никаких действий, кроме увеличения на 1 содержимого счетчика команд. Эта команда используется для отработки временных задержек.

26.3. ДОПОЛНИТЕЛЬНЫЕ ПСЕВДОКОМАНДЫ

При описании псевдокоманды **ORG** отмечалось, что псевдокоманды представляют собой предписание транслятору языка ассемблера. С помощью псевдокоманды **ORG** программист может указать транслятору, с какой позиции памяти будет размещаться программа. Набор псевдокоманд и их представление будут различными для разных ассемблеров, но в любом ассемблере, помимо псевдокоманды **ORG**, существуют еще две псевдокоманды.

Первой из них является псевдокоманда **EQU**. С помощью такой псевдокоманды программист может присвоить значение некоторого выражения используемому в программе символическому обозначению. Эта псевдокоманда имеет следующий формат:

<ИМЯ> EQU <ВЫРАЖЕНИЕ>

Заметим, что после **<ИМЯ>** двоеточие не ставится, несмотря на то что **<ИМЯ>** находится в поле метки. **<ИМЯ>** не является меткой конкретного элемента памяти. Всякий раз, когда в программе будет встречаться обозначение **<ИМЯ>**, транслятор ассемблера вместо него будет подставлять значение выражения, указанного в правой части псевдокоманды **EQU**. Так, в примере с контроллером в начале программы можно ввести следующие псевдокоманды:

IRN	EQU	02H	;НОМЕР ВХОДНОГО ;РЕГИСТРА
ORN	EQU	F2H	;НОМЕР ВЫХОДНОГО ;РЕГИСТРА
LBM	EQU	01H	;МАСКА ДЛЯ ВЫДЕ- ;ЛЕНИЯ МЛАДШЕГО ;РАЗРЯДА

	ORG	3CH	
	JMP	INTR	;ВЕКТОР ПРЕРЫ-
			;ВАНИЯ
	ORG	400H	
INIT:	MVI	A, 0	
	STA	MODE	;УСТАНОВКА ОБЫЧ-
			;НОГО РЕЖИМА
	LXI	SP, 3FFH	;ИНИЦИАЛИЗАЦИЯ
			;УКАЗАТЕЛЯ СТЕКА
	EI		;РАЗРЕШЕНИЕ ПРЕ-
			;РЫВАНИЙ
START:	IN	02H	;ВВОД ЗНАЧЕНИЙ
	MOV	B, A	;СОХРАНЕНИЕ ЗНА-
			;ЧЕНИЙ
	LDA	MODE	;ПРОВЕРКА РЕЖИМА
	ORA	A	;УСТАНОВКА ПРИЗ-
			;НАКОВ
	JNZ	MAN	;ЕСЛИ НЕ НУЛЬ, ТО
			;РУЧНОЙ РЕЖИМ
	LXI	H, TABL	;НАЧАЛЬНЫЙ АДРЕС
			;ТАБЛИЦЫ
			;ПОЛУЧЕНИЕ АДРЕ-
			;СА ЗАПИСИ
	MOV	L, B	
	MOV	A, M	;ПЕРЕСЫЛКА ВЫ-
			;ХОДНОГО ЗНАЧЕНИЯ
	JMP	OUTPUT	
MAN:	IN	04H	;ПЕРЕСЫЛКА ДАН-
			;НЫХ РУЧНОГО ПЕ-
			;РЕКЛЮЧАТЕЛЯ
OUTPUT:	OUT	F2H	
	JMP	START	;ПОВТОРЕНИЕ ПРО-
			;ЦЕССА
INTR:	PUSH	PSW	;СОХРАНЕНИЕ АК-
			;КУМУЛЯТОРА И PSW
	LDA	MODE	

Тогда соответствующие команды в этой программе можно записать в следующем виде:

IN	IRN	;СЧИТЫВАНИЕ ВХОДНЫХ ДАННЫХ
OUT	ORN	;ВЫХОДНЫЕ ДАННЫЕ
ANI	LBM	;СОХРАНЕНИЕ ТОЛЬКО МЛАДШЕГО
		;РАЗРЯДА

Может показаться, что эти дополнительные перекрестные ссылки излишни и бессмысленны, однако на самом деле они существенно облегчают программирование. Обычно псевдокоманды EQU записываются в начале программы. Если в предыдущем примере потребуется изменить номер выходного регистра и сделать его равным СЗН, то единственная замена в псевдокоманде EQU укажет транслятору ассемблера, что повсюду в программе должно быть изменено значение ORN. Выполнение изменений таким способом проще, а кроме того, уменьшается вероятность ошибок при внесении исправлений в текст программы.

Использование символических имен вместо шестнадцатеричных констант делает программу более понятной. Важно также то, что, пользуясь символическими именами, разработку программы можно начинать до того, как будет сконструирована аппаратная часть микроЭВМ и определены способы адресации. Когда же аппаратное обеспечение микроЭВМ спроектировано, останется только определить для псевдокоманд EQU конкретные адреса ячеек памяти и периферийных устройств. Можно отметить, что применение символических имен в целом ускоряет разработку программного обеспечения.

Другой псевдокомандой является псевдокоманда END, которая указывает транслятору ассемблера, что достигнут физический конец программы и больше никаких команд транслировать не надо. Данная псевдокоманда помещается в поле кода операции:

END ;КОНЕЦ ПРОГРАММЫ

26.4. ПРИМЕР РЕАЛИЗАЦИИ КОНТРОЛЛЕРА, ДЕЙСТВУЮЩЕГО В ПОЛУАВТОМАТИЧЕСКОМ РЕЖИМЕ

В контроллере, описанном в гл. 25, может быть предусмотрено участие оператора. В этом случае с помощью входных переключателей может быть осуществлено изменение выходных данных, не предусмотренное программой. Нажатие кнопки, приводящее к прерыванию, будет вызывать чередующуюся смену режимов работы автоматического (программного) и ручного.

Схема алгоритма работы контроллера в полуавтоматическом (интерактивном) режиме представлена на рис. 26.5. Эта схема аналогична той, которая была приведена в предыдущей главе для автоматического режима. После ввода данных выполняется проверка значения переменной с именем MODE. Если эта переменная обозначает ручной режим, то входные данные, задаваемые с помощью переключателей ручного режима, выдаются непосредственно в управляемые устройства.

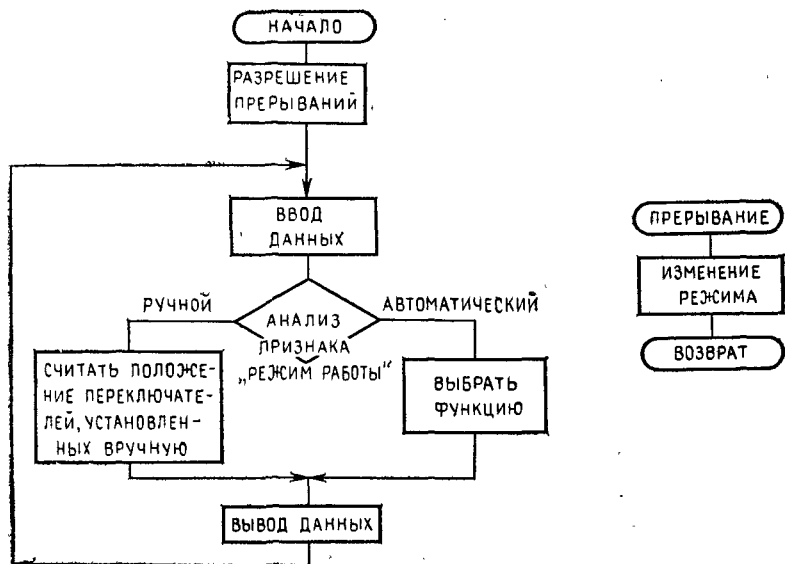


Рис. 26.5. Схема алгоритма работы контроллера в полуавтоматическом (интерактивном) режиме.

Каждое нажатие кнопки вызывает прерывание, и режим работы меняется на противоположный.

Ниже приводится текст программы, реализующей описанный алгоритм. Заметим, что главная программа, за исключением команд проверки значения переменной **MODE**, в целом аналогична представленной в предыдущей главе. В ручном режиме переключатель, подключенный к порту 04H, связывается непосредственно с выходным портом. С помощью программы обработки прерываний после нажатия кнопки прерывания производится изменение всех восьми разрядов переменной **MODE**.

```

ORG 0
JMP INIT      ;НАЧАЛО ПРОГРАММЫ ПОСЛЕ
               ;ВКЛЮЧЕНИЯ ПИТАНИЯ
XRI FFH      ;ИЗМЕНЕНИЕ РЕЖИМА
STA MODE
POP PSW      ;ВОССТАНОВЛЕНИЕ АККУМУЛЯ-
               ;ТОРА И PSW
EI           ;РАЗРЕШЕНИЕ ПЕРЕРЫВАНИЙ
RET          ;ВОЗВРАТ
ORG 500H
TABL: 00H    ;ТАБЛИЦА ЗНАЧЕНИЙ
  
```


;ФУНКЦИИ

02H

02H

06H

02H

02H

02H

06H

02H

02H

02H

06H

06H

0EH

0EH

0FH

MODE: 00

;НУЛЕВОЕ ЗНАЧЕНИЕ СООТВЕТСТВУЕТ АВТОМАТИЧЕСКОМУ РЕЖИМУ, НЕНУЛЕВОЕ—РУЧНОМУ

26.5. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ**Сложение двоичных чисел**

Двоичное сложение можно представить с помощью достаточно простой таблицы.

	0	0	1	1	1-е слагаемое
+	0	1	0	1	2-е слагаемое
	00	01	01	10	Сумма

Сложение многоразрядных двоичных чисел так же, как и десятичных, начинается с младших разрядов. Если в каком-то разряде возникает перенос, то в следующий старший разряд добавляется единица. Например, сложение двух восьмиразрядных чисел выполняется следующим образом:

	00111001	Переносы
	00101101	Операнд
+	10111001	Операнд
	11100110	Сумма

В микропроцессоре 8085 фирмы Intel, как и в большинстве вычислительных машин, значение переноса, сформированного в старшем разряде, запоминается в бите переноса регистра флажков.

При сложении двоичных чисел без знака появление единицы переноса в регистре флажков указывает, что результат операции сложения чрезмерно велик и не может быть представлен имеющимися в распоряжении разрядами регистра. Это явление называется **переполнением**. Очевидно, что по значению бита переноса регистра флажков можно определить факт переполнения. Кроме того, значение переноса, находящееся в регистре флажков, может прибавляться в следующий старший разряд результата в том случае, если результат хранится в нескольких байтах.

Аналогичная таблица сложения составляется и для чисел, представленных в дополнительном коде. Значение бита переноса регистра флажков также прибавляется к следующему старшему разряду результата, если для представления результата используется более чем один байт. Однако в этом случае бит переноса будет определять не переполнение, поскольку при сложении двух чисел с разными знаками переполнения возникнуть не может. Другими словами, разрядность регистра, куда записывается результат операции, всегда будет достаточной для представления суммы.

0111 1111 (127, наибольшее положительное
число)

1111 1111 (—1, наименьшее по модулю)
отрицательное число)

Перенос = 1 0111 1110 (126, правильный ответ)

В данном примере получающийся результат является наибольшим возможным при сложении чисел с разными знаками. Как видно, переполнения не возникает.

1000 0000 (—128, наибольшее по модулю
отрицательное число)

0000 0000 (0, наименьшее положительное
число)

Перенос = 0 1000 0000 (—128, правильный ответ)

В этом примере результат является наибольшим по модулю отрицательным числом, которое можно получить при сложении чисел с разными знаками. И здесь переполнения не возникает.

Другая картина наблюдается при сложении чисел с одинаковыми знаками в дополнительном коде; в этом случае может

возникнуть переполнение. При попытке представить числа, большие, чем +127, и меньшие, чем -128, восемью разрядами окажется, что результат изменит свой знак. Так как сумма двух положительных чисел всегда должна быть положительной, а сумма двух отрицательных чисел — отрицательной, то на возникновение переполнения покажет перемена знака результата. Если оба слагаемых имеют одинаковые знаки, то результат должен иметь тот же знак; в противном случае можно говорить, что произошло переполнение.

$$\begin{array}{r} 1000 \ 0000 \ (-128) \\ 1111 \ 1111 \ (-1) \\ \hline \end{array}$$

Перенос = 1 0111 1111 (+127, неверный ответ)

В данном примере результат сложения двух отрицательных чисел получился положительным. Возникло переполнение.

$$\begin{array}{r} 0111 \ 1111 \ (+127) \\ 0000 \ 0001 \ (+1) \\ \hline \end{array}$$

Перенос = 0 1000 0000 (-128, неверный ответ)

В этом примере результат сложения двух положительных чисел получился отрицательным. Также возникло переполнение.

Команды ADD и ADI

Двоичное представление

	ADD	ADI
Первый байт	10000SSS	11000110
Второй байт	Отсутствует	DATA 8

SSS принимает следующие значения: для В—000, для С—001, для D—010, для E—011, для H—100, для L—101, для аккумулятора—111, для M—110.

Формат на языке ассемблера

(LABEL:) ADD SRC

(LABEL:) ADI DATA

Регистр флажков. Все биты регистра флажков изменяются и имеют обычное назначение.

По командам **ADD** и **ADI** к содержимому аккумулятора прибавляется значение указанного операнда и результат остается в аккумуляторе. Для команды **ADD** операндом является

содержимое произвольного регистра, а также содержимое области памяти, адрес которой определяется регистровой парой. Команда непосредственного сложения **ADI** служит для прибавления восьмиразрядного операнда к содержимому аккумулятора. Изменяются все биты регистра флажков. Бит переноса содержит значение переноса из самого старшего восьмого разряда. В бит дополнительного переноса заносится значение переноса из четвертого разряда в пятый. Биты флажков нуля, знака и четности принимают значения, зависящие от результата операции сложения.

```

STRT:  MVI  A, 10D      ;ЗАНЕСЕНИЕ В АККУМУЛЯТОР
                                   ;ЧИСЛА 10
        MVI  E, 5D        ;ЗАНЕСЕНИЕ В РЕГИСТР D
                                   ;ЧИСЛА 5
        LXI  H, DATA      ;ПАРА РЕГИСТРОВ H И L
                                   ;СОДЕРЖИТ АДРЕС ПОЛЯ
        ADD  E              ;СЛОЖЕНИЕ СОДЕРЖИМОГО
                                   ;АККУМУЛЯТОРА И РЕГИСТРА E
        ADI  2              ;ПРИБАВЛЕНИЕ ЧИСЛА 2
                                   ;К СОДЕРЖИМОМУ АККУМУ-
                                   ;ЛЯТОРА
        ADD  M              ;ПРИБАВЛЕНИЕ СОДЕРЖИ-
                                   ;МОГО ПОЛЯ
        STA  RESULT        ;ЗАПИСЬ РЕЗУЛЬТАТА
        .
        .
DATA:  4D                ;ЧИСЛО 4
RESULT: 0                ;ПОЛЕ РЕЗУЛЬТАТА

```

После выполнения данной программы в поле памяти **RESULT** будет записано число **21D**.

Команды **ADC** и **ACI**

Двоичное представление

	ADD	ACI
Первый байт	10001SSS	11001110
Второй байт	Отсутствует	DATA 8

SSS принимает следующие значения: для В—000, для С—001, для D—010, для E—011, для H—100, для L—101, для аккумулятора—111, для M—110.

*Формат на языке ассемблера***(LABEL:) ADC SRC****(LABEL:) ACI DATA**

Регистр флажков. Все биты регистра флажков изменяются и имеют обычное назначение.

Команда сложения с переносом (**ADC**) и команда прибавления значения второго операнда с переносом (**ACI**) аналогичны командам **ADD** и **ADI**. По командам **ADC** и **ACI** к младшему значащему разряду результата прибавляется также значение бита переноса из регистра флажков. Таким образом, если бит переноса равен 1, то результат увеличивается на 1. Например, если бит переноса после выполнения команды **ADD** равен 1, то с помощью следующей команды **ADC** значение переноса из старшего разряда предыдущего результата прибавляется к младшему разряду нового результата. Таким образом обеспечивается распространение переноса от одного байта к другому при выполнении операции сложения многобайтовых чисел. Ниже приводится пример программы, выполняющей сложение 16-разрядных чисел.

```

STRT: LDA LSG ;МЛАДШИЙ БАЙТ ПЕРВОГО
          ;ОПЕРАНДА
      MOV B, A ;ЗАПОМИНАНИЕ МЛАДШЕГО
          ;БАЙТА
      LDA LSH ;МЛАДШИЙ БАЙТ ВТОРОГО
          ;ОПЕРАНДА
      ADD B ;СЛОЖЕНИЕ МЛАДШИХ БАЙТОВ
          ;ОПЕРАНДОВ И ФИКСАЦИЯ ПЕРЕ-
          ;НОСА В РЕГИСТРЕ ФЛАЖКОВ
      STA LSA ;ЗАПОМИНАНИЕ МЛАДШЕГО
          ;БАЙТА РЕЗУЛЬТАТА
      LDA MSG ;СТАРШИЙ БАЙТ ПЕРВОГО
          ;ОПЕРАНДА
      MOV B, A ;ЗАПОМИНАНИЕ СТАРШЕГО БАЙТА
      LDA MSH ;СТАРШИЙ БАЙТ ВТОРОГО
          ;ОПЕРАНДА
      ADC B ;СЛОЖЕНИЕ СТАРШИХ БАЙТОВ
          ;ОПЕРАНДОВ И ЗНАЧЕНИЯ ПЕРЕ-
          ;НОСА ИЗ МЛАДШЕГО БАЙТА
          ;РЕЗУЛЬТАТА
      STA MSA ;ЗАПОМИНАНИЕ СТАРШЕГО БАЙТА
          ;РЕЗУЛЬТАТА

```

Вычитание двоичных чисел

Двоичное вычитание также можно представить с помощью достаточно простой таблицы.

1	1	0	0	Уменьшаемое
0	1	0	1	Вычитаемое
1	0	0	1	Разность
0	0	0	1	Заем

Вычитание многоразрядных двоичных чисел выполняется так же, как и многоразрядных десятичных чисел, начиная с младших разрядов.

Всякий раз, когда требуется вычесть 1 из 0, из следующего старшего разряда занимается 1. При этом можно считать, что число, составленное из разрядов заема, вычитается из уменьшаемого.

$$\begin{array}{r}
 0110 \ 1000 \text{ Заем} \\
 1101 \ 0110 \text{ Уменьшаемое} \\
 - 0110 \ 1010 \text{ Вычитаемое} \\
 \hline
 0110 \ 1100 \text{ Разность}
 \end{array}$$

Здесь в четвертом разряде выполняется вычитание 1 из 0. Если из пятого разряда занять 1, то получим операцию вычитания 1 из 10, что в результате дает 1. Таким образом, четвертый разряд разности равен 1. Занятая единица вычитается из пятого разряда уменьшаемого, что дает операцию вычитания 0 из 0, результат которой равен 0. Таким образом, пятый разряд разности равен 0.

Если при выполнении операции вычитания в самом старшем значащем разряде требуется занять единицу из следующего разряда, то бит переноса регистра флажков устанавливается в 1. Хотя в указанном случае в регистре флажков используется бит переноса, не следует полагать, что перенос идентичен заему 1 из старшего разряда. Так, при реализации операции вычитания с помощью рассмотренной команды вычитания бит переноса (на самом деле разряд заема) принимает значение, противоположное значению бита переноса в случае выполнения той же операции, но путем прибавления дополнительного кода вычитаемого. Ниже показано, как можно произвести операцию вычитания, используя дополнительный код вычитаемого. Перенос

из старшего значащего разряда противоположен заему единицы в операции вычитания.

	0110	1010	Вычитаемое
	1001	0101	Обратный код вычитаемого
+	1001	0110	Дополнительный код вычитаемого
	1101	0110	Уменьшаемое
	0110	1100	Разность
	1001	0110	Перенос

Если при выполнении операций вычитания двоичных чисел без знака в самом старшем значащем разряде требуется занять единицу, это свидетельствует о том, что вычитаемое больше, чем уменьшаемое. Результат в таком случае должен быть отрицательным, но отрицательное значение не может быть представлено числом без знака. Поэтому значение бита переноса в регистре флажков, равное 1, указывает на переполнение в данной операции вычитания.

При выполнении операции вычитания чисел с одинаковыми знаками, представленных в дополнительном коде, переполнения никогда не возникает.

$$\begin{array}{r} 0111\ 1111 \text{ (127, наибольшее положительное число)} \\ - 0000\ 0000 \text{ (0, наименьшее положительное число)} \\ \hline \end{array}$$

Заем = 0 0111 1111 (127, правильный ответ)

В данном примере получающийся результат является наибольшим при вычитании чисел с одинаковыми знаками. Как видно, переполнения не возникает.

$$\begin{array}{r} 1000\ 0000 \text{ (-128, наибольшее по модулю отрицательное число)} \\ - 1111\ 1111 \text{ (-1, наименьшее по модулю отрицательное число)} \\ \hline \end{array}$$

Заем = 1 1000 0001 (-127, правильный ответ)

Результат данного примера является наибольшим по модулю отрицательным числом, получающимся при вычитании чисел с одинаковыми знаками. И здесь переполнения тоже не происходит.

При выполнении операции вычитания чисел с разными знаками, представленных в дополнительном коде, переполнение возможно. Результат должен иметь тот же знак, что и уменьшаемое; в противном случае возникает переполнение.

$$\begin{array}{r} 0111\ 1111 \text{ (+127)} \\ - 1111\ 1111 \text{ (-1)} \\ \hline \end{array}$$

Заем = 1 1000 0000 (-128, неверный ответ)

Правильный ответ равен $+128$. Результат оказался отрицательным потому, что значение $+128$ не может быть представлено восемью разрядами. Возникло переполнение.

$$\begin{array}{r} - 1000\ 0000\ (-128) \\ - 0000\ 0001\ (+1) \\ \hline \text{Заем} = 0 \quad 0111\ 1111\ (+127, \text{ неверный ответ}) \end{array}$$

Правильный ответ равен -129 . Результат оказался положительным потому, что значение -129 не может быть представлено восемью разрядами. Возникло переполнение.

Команды SUB и SUI

Двоичное представление

	SUB	SUI
Первый байт	10010SSS	11010110
Второй байт		

SSS принимает следующие значения: для В—000, для С—001, для D—010, для E—011, для H—100, для L—101, для M—110, для A—111.

Формат на языке ассемблера

(LABEL:) SUB SRC

(LABEL:) SUI DATA

Регистр флажков. Все биты регистра флажков изменяются и все, кроме бита переноса, имеют обычное назначение. Бит переноса устанавливается в 1, если при выполнении операции вычитания происходит заем.

С помощью команд вычитания (**SUB**) и вычитания непосредственных данных (**SUI**) выполняется операция вычитания данных из содержимого аккумулятора и запоминание результата в аккумуляторе. Вычитаемым для команды **SUB** может быть содержимое любого регистра или поля памяти, адрес которого определяется регистровой парой. Вычитаемым для команды **SUI** являются данные, размещенные во втором байте команды. Если в восьмом разряде осуществляется заем, то бит переноса регистра флажков устанавливается в 1. Однако бит вспомогательного переноса устанавливается в 1, если происходит перенос из четвертого разряда в пятый и нет заема в третьем разряде.

Команды SBB и SBI*Двоичное представление*

	SBB	SBI
Первый байт	10011SSS	11011110
Второй байт		DATA 8

SSS принимает следующие значения: для В—000, для С—001, для D—010, для E—011, для H—100, для L—101, для аккумулятора—111, для M—110.

*Формат на языке ассемблера***(LABEL:) SBB SRC****(LABEL:) SBI DATA**

Регистр флажков. Все биты регистра флажков изменяются и все, кроме бита переноса, имеют обычное назначение. Бит переноса устанавливается в 1, если при выполнении операции вычитания в старшем разряде происходит заем.

Действие команд вычитания с заемом **SBB** и вычитания непосредственных данных с заемом **SBI** аналогично действию команд **SUB** и **SUI**. Но если перед выполнением команды **SBB** или **SBI** бит переноса в регистре флажков равен 1, то из результата операции вычитания дополнительно вычитается 1. Бит переноса регистра флажков устанавливается в 1, если производится заем при выполнении команды вычитания. Поэтому указанные команды целесообразно использовать для реализации операции вычитания многобайтовых чисел. Следующая программа представляет собой пример операции вычитания двух 16-разрядных двоичных чисел **G** и **H**.

```

STRT: LDA  LSG ;МЛАДШИЙ БАЙТ ВЫЧИТАЕМОГО
      MOV  B,A ;ЗАПОМИНАНИЕ МЛАДШЕГО
           ;БАЙТА
      LDA  LSH ;МЛАДШИЙ БАЙТ УМЕНЬШАЕМОГО
      SUB  B   ;ВЫЧИТАНИЕ МЛАДШИХ БАЙТОВ
           ;ЧИСЕЛ, ЗАПОМИНАНИЕ ПРИЗНАКА
           ;ЗАЕМА
      STA  LSA ;ЗАПОМИНАНИЕ МЛАДШЕГО
           ;БАЙТА РЕЗУЛЬТАТА
      LDA  MSG ;СТАРШИЙ БАЙТ ВЫЧИТАЕМОГО
      MOV  B,A ;ЗАПОМИНАНИЕ СТАРШЕГО БАЙТА

```

LDA MSN ;СТАРШИЙ БАЙТ УМЕНЬШАЕМОГО
SBB B ;ВЫЧИТАНИЕ С УЧЕТОМ ЗАЕМА
;В МЛАДШЕМ БАЙТЕ
STA MSA ;ЗАПОМИНАНИЕ СТАРШЕГО БАЙТА
;РЕЗУЛЬТАТА

Команда DAD

Двоичное представление

DAD	
Первый байт	00RP1001
RP принимает следующие значения: для В—00, для D—01, для H—10, для указателя стека (SP)—11.	

Формат на языке ассемблера

(LABEL:) DAD B
(LABEL:) DAD D
(LABEL:) DAD H
(LABEL:) DAD SP

Регистр флажков. Бит переноса устанавливается в 1, если при выполнении операции сложения возникает перенос из старшего разряда самого старшего байта операнда. Другие разряды регистра флажков не изменяются.

Операция сложения 16-разрядных чисел используется в основном для арифметических действий с адресами и обычными численными данными. Эти операции достаточно просто реализуются с помощью команды **DAD** сложения операндов двойной длины. Команда **DAD** обеспечивает сложение содержимого указанной регистровой пары с 16-разрядным числом, записанным в паре регистров **H** и **L**. Результат запоминается в регистрах **H** и **L**. В регистре флажков меняется только бит переноса, в который заносится признак переноса из шестнадцатого значащего разряда числа.

Кроме выполнения операции сложения 16-разрядных чисел, с помощью команды **DAD** реализуются две другие важные функции. Так, по команде **DAD H** производится сложение содержимого пары регистров **H** и **L** с этой же регистровой парой. Это эквивалентно операции сдвига 16-разрядного кода числа влево на один разряд. Таким образом, команда **DAD H** пред-

ставляет собой команду сдвига влево на один разряд содержимого пары регистров **H** и **L**.

Единственным средством, позволяющим получить значение содержимого указателя стека, является команда **DAD SP**. С ее помощью в программе можно выполнить проверку содержимого указателя стека с целью определить, имеется ли в стеке еще место для занесения туда данных. В следующем примере приводится программа, в которой значение содержимого указателя стека сравнивается с заданным предельным адресом, хранящимся в некотором поле памяти.

STRT:	LDA	LIMIT	;МЛАДШИЙ БАЙТ ПРЕДЕЛЬ-
			;НОГО АДРЕСА
	CMA		;ПОЛУЧЕНИЕ ОБРАТНОГО
			;КОДА МЛАДШЕГО БАЙТА
	MOV	L, A	;ЗАПОМИНАНИЕ В РЕГИСТРЕ L
	GDA	LIMIT+1	;СТАРШИЙ БАЙТ ПРЕДЕЛЬ-
			;НОГО АДРЕСА
	CMA		;ПОЛУЧЕНИЕ ОБРАТНОГО КОДА
			;СТАРШЕГО БАЙТА
	MOV	H, A	;ЗАПОМИНАНИЕ В РЕГИСТРЕ H
	INX	H	;ПРИБАВЛЕНИЕ 1 ДЛЯ ПОЛУ-
			;ЧЕНИЯ ДОПОЛНИТЕЛЬНОГО
			;КОДА
	DAD	SP	;ПРИБАВЛЕНИЕ СОДЕРЖИ-
			;МОГО УКАЗАТЕЛЯ СТЕКА
	JNC	OFLW	;ПЕРЕХОД, ЕСЛИ СОДЕРЖИМОЕ
			;УКАЗАТЕЛЯ СТЕКА БОЛЬШЕ
			;ПРЕДЕЛЬНОГО ЗНАЧЕНИЯ
			;АДРЕСА
	NOP		;В ПРОТИВНОМ СЛУЧАЕ — ВЫ-
			;ПОЛНИТЬ ОПЕРАЦИЮ ЗАНЕ-
			;СЕНИЯ В СТЕК

В данной программе выполняется сложение содержимого указателя стека с дополнительным кодом значения адреса, записанного в поле памяти с именем **LIMIT**. Поскольку по ходу загрузки стека содержимое указателя стека возрастает, признаком переполнения стека является превышение предельного значения содержимым указателя стека. Разность между значением указателя стека и значением в поле **LIMIT** положительна, если значение указателя стека больше или равно предельному

значению. Поэтому при использовании операции вычитания бит заема должен равняться 0. Но так как вычитание заменяется сложением чисел в дополнительном коде, то значение переноса противоположно значению заема. Таким образом, бит переноса равен 1 тогда, когда содержимое указателя стека больше или равно содержимому поля **LIMIT**. Значение бита переноса равно 0, если объем стека превышает предельное значение.

Команды **CMP** и **CPI**

Двоичное представление

	CMP	CPI
Первый байт	10111SSS	11111110
Второй байт	Отсутствует	DATA 8

SSS принимает следующие значения: для В—000, для С—001, для D—010, для E—011, для H—100, для L—101, для аккумулятора—111, для M—110.

Формат на языке ассемблера

(LABEL:) CMP SRC

(LABEL:) CPI DATA

Регистр флажков. Все биты регистра флажков изменяются, и все, кроме бита переноса, имеют обычное назначение. Бит переноса устанавливается в 1, если при выполнении операции вычитания происходит заем.

С помощью команд сравнения **CMP** и сравнения непосредственных данных **CPI** выполняется операция сравнения содержимого указанного байта с содержимым аккумулятора. Сравнимыми данными могут быть содержимое любого регистра или содержимое поля памяти, адрес которого определяется заданной регистровой парой. По команде **CPI** производится сравнение с непосредственными данными, записанными во втором байте команды.

Сравнение выполняется за счет вычитания указанного байта данных из содержимого аккумулятора. Биты регистра флажков принимают те же значения, что и для команды **SUB**. Результат операции вычитания в данном случае нигде не запоминается. Значения сравниваемых данных не изменяются. Полученные значения битов регистра флажков могут быть использованы в программе при выполнении операции условного перехода.

Если бит нуля в регистре флажков равен 1, то это означает, что оба операнда равны; в противном случае они не равны. Бит

знака в регистре флажков в отсутствие переполнения может использоваться для относительного сравнения значений операндов. В этом случае, если бит знака равен 1, что соответствует отрицательному значению результата, то указанный байт данных превосходит содержимое аккумулятора. Если бит знака равен 0, то содержимое аккумулятора больше или равно указанному байту данных. Недостатком такой реализации операции сравнения является то, что всегда требуется проверять, не возникло ли переполнения. Без учета переполнения операция сравнения может быть выполнена с помощью анализа состояния бита переноса регистра флажков, поскольку этот бит будет принимать значение бита заема в самом старшем значащем разряде для используемой операции вычитания.

Если оба операнда имеют одинаковые знаки и содержимое аккумулятора больше или равно указанному байту данных, то заем не производится и бит переноса равен 0. Если бит переноса равен 1, что свидетельствует о наличии заема, то указанный байт данных превосходит содержимое аккумулятора. Если знаки операндов разные, смысл значений бита переноса меняется на противоположный. Обобщенно результаты операции сравнения можно выразить следующим образом:

Знаки операндов	Биты регистра флажков	Значение
Любые	Нуль = 1	(Аккумулятор) = Данные
Любые	Нуль = 0	(Аккумулятор) \neq Данные
Одинаковые	Перенос = 1	(Аккумулятор) < Данные
Одинаковые	Перенос = 0	(Аккумулятор) \geq Данные
Разные	Перенос = 1	(Аккумулятор) \geq Данные
Разные	Перенос = 0	(Аккумулятор) < Данные

Если сравниваемые числа достаточно малы, то для выполнения операции сравнения необходимо проверить только бит знака. Часто значения сравниваемых чисел лежат в таком диапазоне, что переполнений не возникает.

Команда DAA

Двоичное представление

DAA	
Первый байт	00100111

Формат на языке ассемблера

(LABEL:) DAA

Регистр флажков. Все биты регистра флажков изменяются и имеют обычное назначение.

Команда **DAA** коррекции десятичных чисел, записанных в аккумулятор, используется совместно с другими командами, производящими арифметические операции над числами, представленными в двоично-десятичном коде. Если два числа, представленные в двоично-десятичном коде и изменяющиеся в диапазоне от 0 до 9, сложить с помощью команд двоичной арифметики, то представление результата в двоично-десятичном коде может оказаться неверным. Например:

Случай 1	+	0001	(1)	
		0011	(3)	
		0100	(4)	
		0001	(1)	Допустимый двоично-десятичный код
Случай 2	+	1001	(9)	
		1010	(A)	
		0111	(7)	
Случай 3	+	1001	(9)	
		0000	(0)	
Перенос = 1		0000	(0)	Допустимый двоично-десятичный код, но неверный результат операции

Приведенные примеры характеризуют все три возможных случая. Видно, что получаемый результат может быть правильным и выражаться допустимым двоично-десятичным кодом. Но результат вычислений может оказаться и шестнадцатеричным числом в диапазоне от A до F (что недопустимо для двоично-десятичного представления чисел). Кроме того, может получиться двоично-десятичное представление чисел от 0 до 8, но результат будет неверным (всегда возникает перенос).

С помощью команды **DAA** в последних двух случаях производится соответствующая коррекция, дающая правильный ответ в двоично-десятичном коде. После сложения двух двоично-десятичных чисел, выполненного с помощью команды **ADD**, по команде **DAA** осуществляется коррекция результата, состоящая из двух шагов.

1. Если младший десятичный разряд числа больше 9 (случай 2) или значение вспомогательного переноса в этом байте равно 1 (случай 3), то к содержимому аккумулятора прибавляется 6. Тогда в примере для случая 2 значение младшего значащего десятичного разряда в двоичном представлении будет равно 0000, а в следующий старший значащий разряд будет прибавлена 1. В примере для случая 3 значение младшего значащего десятичного разряда станет равным 6. В примере для случая 2 распространение единицы переноса в следующий раз-

ряд при замене шестнадцатеричного числа А числом 0 является, по сути дела, прибавлением числа 10 ко всему десятичному числу. В примере для случая 3 единица переноса в следующий десятичный разряд возникает в результате выполнения обычной операции двоичного сложения, так что при замене числа 0 число 6 дополнительного переноса не требуется.

2. После этого с помощью команды **DAA** производится коррекция старшего значащего десятичного разряда. Если значение в этом разряде больше 9 (случай 2) или если значение переноса равно 1 (случай 3), то к этому разряду прибавляется число 6. Значение переноса из восьмого разряда соответствующего байта запоминается, как обычно, в бите переноса регистра флажков. Остальные биты регистра флажков принимают стандартные значения в соответствии с результатом операции.

После выполнения команды **DAA** в аккумуляторе будет записан допустимый двоично-десятичный код суммы двух двоично-десятичных чисел. Если сумма превосходит 99, то бит переноса будет равен 1, что соответствует переполнению и появлению единицы переноса в следующем более старшем десятичном разряде. Если суммируемые операнды имеют более двух десятичных разрядов, то для выполнения такой операции сложения можно использовать команду **ADC**. Следует отметить, что после команды вычитания команда **DAA** не используется. Поэтому операция десятичного вычитания должна выполняться с помощью прибавления значения вычитаемого в десятичном дополнительном коде. Десятичный дополнительный код для отрицательных двоично-десятичных чисел аналогичен дополнительному коду двоичных чисел. В приводимом ниже примере представлена программа, реализующая операцию вычитания указанным способом.

STRT:	LXI	D,MIN	;АДРЕС УМЕНЬШАЕМОГО
	LXI	H,SBTR	;АДРЕС ВЫЧИТАЕМОГО
	MVI	C,2	;4 ЦИФРЫ ИЛИ 2 БАЙТ
	STC		;ЗАЕМ ОТСУТСТВУЕТ
LOOP:	MVI	99H	;ДВОИЧНО-ДЕСЯТИЧНОЕ
			;ЧИСЛО 99
	ACI	0	;ПРИБАВЛЕНИЕ ЗНАЧЕНИЯ
			;ПЕРЕНОСА
	SUB	M	;ДОПОЛНЕНИЕ ВЫЧИТАЕМОГО
	XCHG		;ПОЛУЧЕНИЕ АДРЕСА УМЕНЬ-
			;ШАЕМОГО
	ADD	M	;ПРИБАВЛЕНИЕ УМЕНЬШАЕ-
			;МОГО

DAA		;КОРРЕКЦИЯ ДЕСЯТИЧНОГО ;РЕЗУЛЬТАТА
MOV	M,A	;ЗАПОМИНАНИЕ РЕЗУЛЬТАТА
XCHG		;ПОЛУЧЕНИЕ АДРЕСА ВЫЧИ- ;ТАЕМОГО
DCR	C	;ВСЕ ЛИ БАЙТЫ ОБРАБОТАНЫ?
JZ	DONE	;ДА—ВЫПОЛНЕНИЕ ЗАКОН- ;ЧЕНО
INX	D	;НЕТ—ПОЛУЧЕНИЕ АДРЕСА ;СЛЕДУЮЩЕГО БАЙТА
INX	H	;СЛЕДУЮЩИЙ БАЙТ
JMP	LOOP	;ПОЛУЧЕНИЕ СЛЕДУЮЩИХ ;ДВУХ ДЕСЯТИЧНЫХ ЦИФР

26.6. ДОПОЛНИТЕЛЬНЫЕ КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ

Команды XCHG и XTHL

Двоичное представление

	XCHG	XTHL
Первый байт	11101011	11100011

Формат на языке ассемблера

(LABEL:) XCHG

(LABEL:) XTHL

Регистр флажков. Не изменяется.

С помощью команды **XCHG** осуществляется обмен содержимым пары регистров **D** и **E** и пары регистров **H** и **L**. Для задания операнда во многих командах логических или арифметических операций часто используется адресация посредством пары регистров. Причем этим целям служит именно пара регистров **H** и **L**. Определенные удобства предоставляет применение команды **XCHG**: для обращения к другому операнду адрес этого операнда можно подготовить в паре регистров **D** и **E**, а затем с помощью команды **XCHG** перенести этот адрес в регистры **H** и **L**. Ниже приводится текст программы, иллюстрирующей применение команды **XCHG**, в которой выполняется прибавление постоянного числа к элементам, далее из этих элементов вычитаются соответствующие элементы другого массива, а результат запоминается в поле первого массива,

STRT:	LXI	D,AR2	;ВТОРОЙ МАССИВ
	LXI	H,AR1	;ПЕРВЫЙ МАССИВ
	MVI	B,SIZE	;РАЗМЕР МАССИВОВ
LOOP:	MVI	A,CONST	;ПОСТОЯННОЕ ЧИСЛО
	ADD	M	;ПРИБАВЛЕНИЕ ПОСТОЯН-
			;НОГО ЧИСЛА К ЭЛЕМЕНТАМ
			;ПЕРВОГО МАССИВА
	XCHG		;ИЗМЕНЕНИЕ АДРЕСОВ
	SUB	M	;ВЫЧИТАНИЕ ЭЛЕМЕНТОВ
			;ВТОРОГО МАССИВА
	XCHG		;ВОССТАНОВЛЕНИЕ АДРЕСА
			;ПЕРВОГО МАССИВА
	MOV	M,A	;ЗАПОМИНАНИЕ РЕЗУЛЬТАТА
			;В ПЕРВОМ МАССИВЕ
	INX	D	;СЛЕДУЮЩИЙ ЭЛЕМЕНТ
			;ВТОРОГО МАССИВА
	INX	H	;СЛЕДУЮЩИЙ ЭЛЕМЕНТ
			;ПЕРВОГО МАССИВА
	DCR	B	;УМЕНЬШЕНИЕ СОДЕРЖИ-
			;МОГО СЧЕТЧИКА НА 1,
			;ЕСЛИ ЗНАЧЕНИЕ СЧЕТЧИКА
			;НЕ РАВНО НУЛЮ
	JNZ	LOOP	;ПРОДОЛЖАТЬ ОБРАБОТКУ

С помощью команды **XTHL** обмена содержимым регистров **H** и **L** и двух верхних позиций стека можно изменять текущее значение адреса возврата и запоминать его первоначальное значение.

Команда **SPHL**

Двоичное представление

SPHL	
Первый байт	11111001

Формат на языке ассемблера

(LABEL:) SPHL

Регистр флажков. Не изменяется.

Посредством команды **SPHL** осуществляется пересылка содержимого регистров **H** и **L** в указатель стека. Подобно команде **LXI SP, DATA**, эта команда может использоваться для начальной установки указателя стека. Правда, с помощью команды **LXI** в указатель стека заносится заранее определенная константа, тогда как по команде **SPHL** в указатель стека можно записать любое значение, находящееся в регистрах **H** и **L**.

26.7. ПРОГРАММИРУЕМЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Умножение

В систему команд некоторых микропроцессоров, в том числе и серии 8085 фирмы Intel, не входят команды умножения или деления. Эти операции выполняются с помощью подпрограмм, написанных программистом. Рассмотрим сначала умножение двоичных чисел без знака. Но перед этим разберем пример умножения двух десятичных чисел:

$$\begin{array}{r}
 \times 483 \text{ Множимое} \\
 \underline{24 \text{ Множитель}} \\
 1932 \text{ Частичное произведение} \\
 966 \text{ Частичное произведение} \\
 \hline
 11592 \text{ Произведение}
 \end{array}$$

Умножение начинается с самого правого разряда множителя. Воспользуемся таблицей десятичного умножения и применим правила распространения переноса, в результате получим первое частичное произведение. Следующий разряд множителя порождает второе частичное произведение, которое сдвигается влево на одну позицию относительно первого частичного произведения, что эквивалентно умножению на 10. Таким же образом продолжается формирование всех частичных произведений, соответствующих остальным разрядам множителя. Чтобы получить искомое произведение, нужно просуммировать все частичные произведения.

Итак, первое, что необходимо сделать для реализации операции умножения двоичных чисел,— это заготовить таблицу двоичного умножения:

	0	0	1	1	Множимое
	0	1	0	1	Множитель
×	<hr/>				
	0	0	0	1	Произведение

Перемножим два четырехразрядных числа без знака:

$$\begin{array}{r}
 \times 1011 \quad (11) \\
 0101 \quad (5) \\
 \hline
 1011 \\
 0000 \\
 1011 \\
 0000 \\
 \hline
 0110111 \quad (55)
 \end{array}$$

Как видно из двоичной таблицы умножения, разряду множителя, равному 1, соответствует частичное произведение, равное значению множимого, а нулевому разряду множителя — частичное произведение, равное 0. Поэтому в зависимости от значений разрядов множителя значение множимого либо прибавляется, либо нет. Запоминать каждое частичное произведение нет необходимости: для вычисления искомого произведения целесообразно частичные произведения складывать друг с другом сразу по мере их получения.

Выполним предыдущий пример по этому правилу:

0000	1011	Множимое
	0101	Множитель
0000	0000	Накопленное частичное произведение

Так как младший разряд множителя равен 1, прибавим значение множимого к частичному произведению. Вместе с тем код множителя сдвинем вправо на один разряд для проверки следующего разряда, а код множимого — влево на один разряд, чтобы подготовить его для сложения с частичным произведением.

0001	0110	Множимое
	0010	Множитель
0000	1011	Частичное произведение

Поскольку следующий разряд множителя равен 0, то выполняются только операции сдвига кодов множимого и множителя без прибавления значения множимого к частичному произведению.

0010	1100	Множимое
	0001	Множитель
0000	1011	Частичное произведение

Следующий разряд множителя равен 1, поэтому множимое прибавляется.

0101	1000	Множимое
	0000	Множитель
0011	0111	Частичное произведение

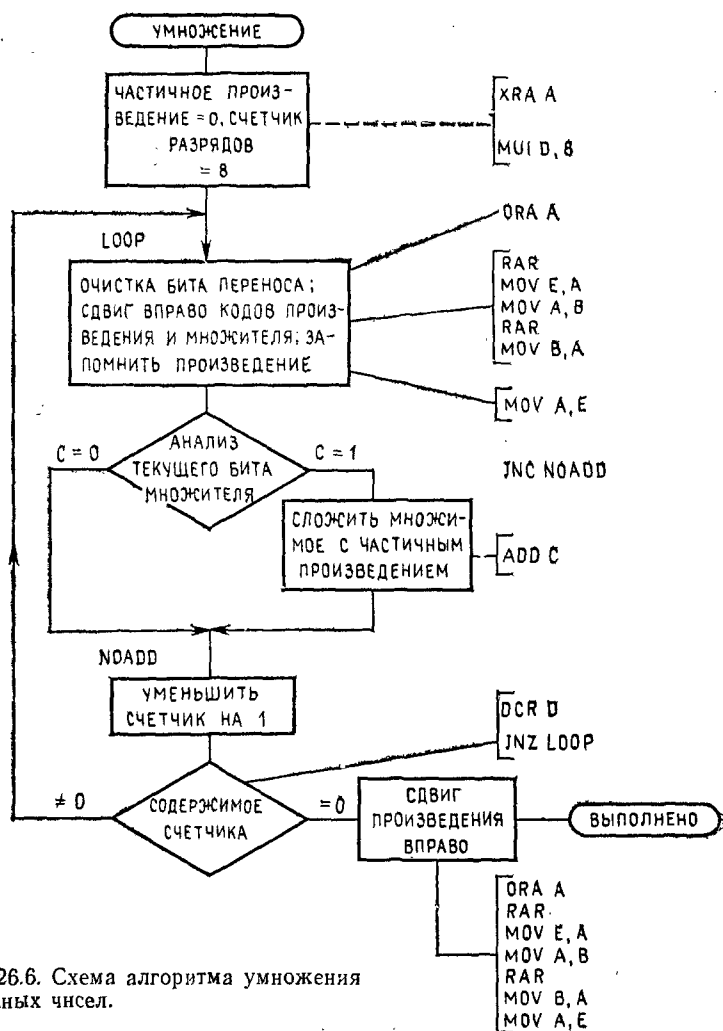


Рис. 26.6. Схема алгоритма умножения двоичных чисел.

Последний разряд множителя равен 0, так что множимое прибавлять не надо.

1011	0000	Множимое
-	0000	Множитель
0011	0111	Произведение

Приведенный алгоритм можно реализовать, однако на практике он не применяется по ряду причин. Во-первых, в указанном алгоритме для хранения четырехразрядного кода множимого требуется 8-разрядный регистр; во-вторых, несмотря на то что

операнды являются 4-разрядными, приходится выполнять сложение 8-разрядных чисел. Поэтому вместо сдвига кода множителя влево целесообразно сдвигать вправо код частичного произведения.

	НАЧАЛО	1011	Множимое
		0101	Множитель
		0000 0000	Произведение (начальное значение)
1. СЛОЖЕНИЕ		1011	2. СДВИГ 1011
		0101	0010
		1011 0000	0101 1000
3. БЕЗ СЛОЖЕНИЯ		1011	4. СДВИГ 1011
		0010	0001
		0101 1000	0010 1100
5. СЛОЖЕНИЕ		1011	6. СДВИГ 1011
		0001	0000
		1101 1100	0110 1110
7 БЕЗ СЛОЖЕНИЯ		1011	8. СДВИГ 1011
		0000	0000
		0110 1110	0011 0111

Заметим, что сдвиг вправо кодов множителя и накопленного частичного произведения выполняется на каждом шаге. Поэтому код множителя целесообразно записывать в младшие разряды, предназначенные для произведения. Тогда, выполняя на каждом шагу сдвиг вправо, можно одновременно выделять для проверки очередной разряд множителя и освобождать слева место для размещения нового значения накопленного произведения.

Алгоритм умножения для микропроцессора 8085 фирмы Intel представлен на рис. 26.6. Проверить значение самого младшего разряда регистра без проведения дополнительных логических операций с маской в микропроцессоре 8085 нельзя. Поэтому для получающегося накопленного произведения и множителя выполняется операция сдвига вправо, в результате чего младший значащий разряд регистра заносится в бит переноса регистра флажков. Проверка же значения бита переноса производится с помощью команды условного перехода. Описанный алгоритм умножения реализуется в следующей программе:

```

;УМНОЖЕНИЕ С ОБЫЧНОЙ ТОЧ-
;НОСТЬЮ
;МНОЖИТЕЛЬ В РЕГИСТРЕ В
;МНОЖИМОЕ В РЕГИСТРЕ С
;РЕЗУЛЬТАТ В РЕГИСТРАХ А И В
;(16 РАЗЯДОВ)
;СЧЕТЧИК РАЗЯДОВ В РЕГИ-
;СТРЕ D
;ИНИЦИАЛИЗАЦИЯ ЧАСТИЧНОГО
;ПРОИЗВЕДЕНИЯ
МРУ: ХРА А

```

	MVI	D,8	;ПРОВЕРКА ВОСЬМИ РАЗРЯДОВ
			;МНОЖИТЕЛЯ
LOOP:	ORA	A	;ОЧИСТКА БИТА ПЕРЕНОСА
	RAR		;СДВИГ ПРОИЗВЕДЕНИЯ ВПРАВО
	MOV	E,A	;ЗАПОМИНАНИЕ ЧАСТИЧНОГО
			;ПРОИЗВЕДЕНИЯ
	MOV	A,B	;ПОЛУЧЕНИЕ МНОЖИТЕЛЯ
	RAR		;ЦИКЛИЧЕСКИЙ СДВИГ
	MOV	B,A	;СОХРАНЕНИЕ МНОЖИТЕЛЯ
	MOV	A,E	;СОХРАНЕНИЕ ЧАСТИЧНОГО
			;ПРОИЗВЕДЕНИЯ
	JNC	NOADD	;ПРОВЕРКА ТЕКУЩЕГО БИТА
			;МНОЖИТЕЛЯ
	ADD	C	;РАЗРЯД МНОЖИТЕЛЯ РАВЕН 1,
			;ПРИБАВЛЕНИЕ МНОЖИМОГО
NOADD:	DCR	D	;УМЕНЬШЕНИЕ СОДЕРЖИМОГО
			;СЧЕТЧИКА НА 1
	JNZ	LOOP	;СЛОЖЕНИЕ НЕ ПРОИЗВОДИТСЯ,
			;ПЕРЕХОД К СЛЕДУЮЩЕМУ
			;ЧАСТИЧНОМУ ПРОИЗВЕДЕНИЮ
	ORA	A	;ОЧИСТКА БИТА ПЕРЕНОСА
	RAR		;ЕЩЕ ОДИН СДВИГ ПРОИЗВЕ-
			;ДЕНИЯ
	MOV	E,A	
	MOV	A,B	
	RAR		
	MOV	B,A	
	MOV	A,E	

Деление

Операция двоичного деления выполняется аналогично делению десятичных чисел. Рассмотрим операцию десятичного деления:

Делимое	478		22	Делитель
	44		21	Частное
	<u>38</u>			
	22			
	<u>16</u>			Остаток

Вначале предпринимается попытка разделить на делитель первую цифру делимого. В данном случае это деление не может быть выполнено, поэтому первая цифра частного устанавливается равной 0. Затем аналогичная попытка деления предпринимается для числа, составленного из первых двух цифр делимого, и т. д. до тех пор, пока такое деление будет возможным. Вычисляется остаток такой операции деления, справа к нему приписывается следующая цифра делимого и опять делается попытка разделить полученное число на делитель. Если деление не выполнимо, то в частное в качестве следующей цифры записывается 0, а к числу, которое не удалось разделить, справа приписывается следующая цифра делимого, и вновь предпринимается попытка деления на делитель. Если это деление возможно, то в частное записывается очередная цифра и вычисляется новое значение остатка.

Деление двоичных чисел производится по аналогичной схеме:

$$\begin{array}{rcl}
 (167) & \begin{array}{cc} 1010 & 0111 \\ 0110 & \end{array} & \left| \begin{array}{cc} 0000 & 0110 \\ 0001 & 1011 \end{array} \right. \begin{array}{l} (6) \\ (27) \end{array} \\
 & \begin{array}{cc} 0100 & 0 \\ 011 & 0 \end{array} & \\
 & \begin{array}{cc} 001 & 011 \\ 0 & 110 \end{array} & \\
 & \begin{array}{cc} 0 & 1011 \\ & 0110 \end{array} & \\
 & 0101 & (5)
 \end{array}$$

Как и для операции умножения, действия с двоичными числами в операции деления достаточно просты. В зависимости от того, чему равен очередной разряд частного (1 или 0), делитель либо вычитается из остатка, либо нет. На самом же деле именно операцию вычитания делителя из делимого можно использовать для определения значения очередного разряда частного. Рассмотрим подробнее этот алгоритм.

Остаток	Делимое и частное
0000 0000	1010 0111

Сдвинем коды остатка и делимого влево и попытаемся вычесть делитель.

$$1. \quad \begin{array}{r} 0000 \ 0001 \ 0100 \ 1110 \\ - 0000 \ 0110 \\ \hline \end{array}$$

Результат получается отрицательным. Поэтому еще раз выполним операцию сдвига и попытаемся снова вычесть делитель. В частное запишем 0. Поскольку на каждом шаге код делимого сдвигается влево, освобождающиеся справа разряды целесообразно использовать для записи частного.

$$\begin{array}{r} 2. \quad 0000 \ 0010 \ 1001 \ 1100 \\ - \quad 0000 \ 0110 \\ \hline \end{array}$$

Результат отрицательный. В частное записывается 0.

$$\begin{array}{r} 3. \quad 0000 \ 0101 \ 0011 \ 1000 \\ - \quad 0000 \ 0110 \\ \hline \end{array}$$

Результат отрицательный. В частное записывается 0.

$$\begin{array}{r} 4. \quad 0000 \ 1010 \ 0111 \ 0000 \\ - \quad 0000 \ 0110 \\ \hline 0000 \ 0100 \end{array}$$

Результат положительный. В частное записывается 1.

$$\begin{array}{r} 5. \quad 0000 \ 1000 \ 1110 \ 0001 \\ - \quad 0000 \ 0110 \\ \hline 0000 \ 0010 \end{array}$$

Результат положительный. В частное записывается 1.

$$\begin{array}{r} 6. \quad 0000 \ 0101 \ 1100 \ 0011 \\ - \quad 0000 \ 0110 \\ \hline \end{array}$$

Результат отрицательный. В частное записывается 0.

$$\begin{array}{r} 7. \quad 0000 \ 1011 \ 1000 \ 0110 \\ - \quad 0000 \ 0110 \\ \hline 0000 \ 0101 \end{array}$$

Результат положительный. В частное записывается 1.

$$\begin{array}{r} 8. \quad 0000 \ 1011 \ 0000 \ 1101 \\ - \quad 0000 \ 0110 \\ \hline 0000 \ 0101 \end{array}$$

Результат положительный. В частное записывается 1.

$$9. \ 0000 \ 1010 \ 0001 \ 1011$$

Итак, просмотрены все разряды делимого и вычислено частное. Для того чтобы определить последний разряд частного, необходимо код остатка сдвинуть дополнительно на один разряд вправо.

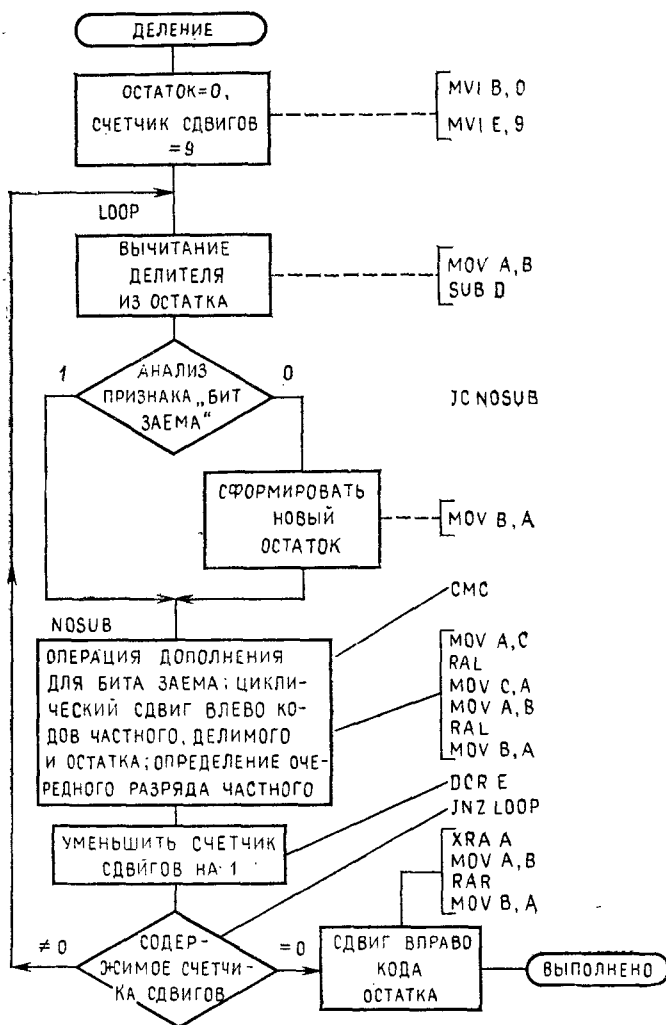


Рис. 26.7. Схема алгоритма деления двоичных чисел.

Подобно тому как в операции умножения для записи множителя и произведения использовался один и тот же регистр, в операции деления один и тот же регистр используется для делимого и частного. На каждом этапе алгоритма деления получается новый разряд частного, а очередной разряд делимого приписывается к коду получающегося остатка. Так как действия выполняются над числами без знака, результат считается

отрицательным, когда значение признака заема равно 1. Это значение записывается в бит переноса регистра флажков. Поскольку очередной разряд частного принимается равным 1, если промежуточный результат положительный, и равным 0, если промежуточный результат отрицательный, то в качестве значения разряда частного можно использовать дополнение бита переноса промежуточного результата. Первая операция сдвига может рассматриваться как результат некоторого дополнительного шага алгоритма деления. Действительно, вычитание делимого из остатка, имеющего нулевое начальное значение, порождает значение разряда частного, равное 0, после чего выполняется указанная операция сдвига влево. В приведенном выше примере этот дополнительный шаг алгоритма деления эквивалентен первому из девяти сдвигов.

На рис. 26.7 представлена схема данного алгоритма. В соответствии с этой схемой может быть написана следующая программа деления:

			;ДЕЛЕНИЕ С ОБЫЧНОЙ ТОЧ-
			;НОСТЬЮ
			;ОСТАТОК В РЕГИСТРЕ В
			;ДЕЛИМОЕ И ЧАСТНОЕ В РЕ-
			;ГИСТРЕ С
			;ДЕЛИТЕЛЬ В РЕГИСТРЕ D
			;СЧЕТЧИК РАЗРЯДОВ В РЕГИ-
			;СТРЕ E
DIV:	MVI	E,9	;9 СДВИГОВ
	MVI	B,0	;ОСТАТОК РАВЕН НУЛЮ
LOOP:	MOV	A,B	;ПОЛУЧЕНИЕ ОСТАТКА
	SUB	D	;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ
	JC	NOSUB	;ОТРИЦАТЕЛЬНЫЙ РЕЗУЛЬТАТ,
			;ЭТОТ РЕЗУЛЬТАТ
			;НЕ ЗАПОМИНАЕТСЯ
	MOV	B,A	;ПОЛОЖИТЕЛЬНЫЙ РЕЗУЛЬ-
			;ТАТ, ЗАПОМИНАНИЕ НОВОГО
			;ОСТАТКА
NOSUB:	CMC		;ДЛЯ НОВОГО РАЗРЯДА
			;ЧАСТНОГО
	MOV	A,C	;ПОЛУЧЕНИЕ ЧАСТНОГО И
			;ДЕЛИМОГО
	RAL		;ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО
	MOV	C,A	;СОХРАНЕНИЕ

MOV	A,B	;ПОЛУЧЕНИЕ ОСТАТКА
RAL		;ВКЛЮЧЕНИЕ НОВОГО РАЗРЯДА
		;ДЕЛИМОГО
MOV	B,A	;СОХРАНЕНИЕ
DCR	E	;УМЕНЬШЕНИЕ СОДЕРЖИМОГО
		;СЧЕТЧИКА
JNZ	LOOP	;ВЫПОЛНЕНИЕ 9 РАЗ
XRA	A	;ОЧИСТКА БИТА ПЕРЕНОСА
MOV	A,B	;ПОЛУЧЕНИЕ ОСТАТКА
RAR		;КОРРЕКЦИЯ ОСТАТКА
MOV	B,A	;СОХРАНЕНИЕ

Умножение и деление чисел со знаком

Один из способов выполнения операций умножения и деления для чисел в дополнительном коде предусматривает независимое определение знака результата и выполнение действий над абсолютными значениями операндов. Для полученного результата проводится знаковая коррекция. Если знаки операндов разные, то результат должен быть отрицательным. Определение знака результата умножения или деления может выполняться с помощью отдельной программы.

Описанный ранее алгоритм умножения чисел без знака может быть использован и в случае отрицательного значения множителя. При этом необходимо только предварительно вычислить абсолютное значение множителя. Если первоначальный знак множителя был отрицательным, то знак результата изменится.

Для выполнения операций умножения и деления чисел со знаком в дополнительном коде существует множество других алгоритмов, но при их реализации на микроЭВМ выигрыш, как правило, получается незначительным или отсутствует вовсе.

Умножение на константы, содержащие малое число единичных разрядов

Часто требуется выполнять операции умножения на константы, содержащие в своем двоичном представлении небольшое количество единичных разрядов. Например, если строка таблицы в постоянной памяти занимает 3 байта, то для определения адреса этой строки ее номер в таблице должен умножаться на 3. В одном из ранее приведенных примеров было показано, что каждую трехбайтную строчку таблицы можно дополнить четвертым «холостым» байтом. Если воспользоваться таким приемом, то для обращения к определенной строке таблицы

потребуется операция умножения на 4, которая может быть реализована посредством операций сдвига влево. Однако если в таблице много строк, то использование дополнительных «холостых» байтов приведет к неэффективному расходованию памяти. С другой стороны, для умножения на 3 или 5 универсальная программа умножения оказывается громоздкой и медленной. При умножении на константы с небольшим числом единичных разрядов целесообразно выполнять только те шаги алгоритма умножения, которые соответствуют единичным разрядам множителя. Если в двоичном представлении множителя содержится всего несколько единиц (как для чисел 3 или 5), то выполнение операции умножения таким способом будет значительно быстрее, чем с помощью обычной программы умножения. В качестве примера приведем программу, в которой реализован описанный выше метод обращения к строкам таблицы, имеющим длину 3 байта.

STRT:	LDA	RTNUM	;НОМЕР ПОДПРОГРАММЫ
	MOV	B,A	;СОХРАНЕНИЕ ИСХОДНОГО
			;ЧИСЛА
	ADD	A	;УМНОЖЕНИЕ НА 2
	ADD	B	;2*RTNUM+RTNUM=3*RTNUM
	LXI	H,JTBL	;НАЧАЛЬНЫЙ АДРЕС ТАБЛИЦЫ
			;ПЕРЕХОДОВ
	MVI	B,0	;СТАРШИЙ БАЙТ СМЕЩЕНИЯ
	MOV	C,A	;МЛАДШИЙ БАЙТ СМЕЩЕНИЯ
	DAD	B	;ПРИБАВЛЕНИЕ СМЕЩЕНИЯ
			;К НАЧАЛЬНОМУ АДРЕСУ
	PCNL		;ПЕРЕХОД ПО ЭТОМУ АДРЕСУ
JTBL:	JMP	RT0	;СТРОКА ТАБЛИЦЫ ДЛИНОЙ
			;3 БАЙТА
	JMP	RT1	
	JMP	RT2	
	.		
	.		
	JMP	RT84	;МАКСИМАЛЬНЫЙ ДЕСЯТИЧ-
			;НЫЙ НОМЕР ПОДПРОГРАММЫ
			;РАВЕН 84

Если требуется, чтобы таблица содержала более 84 строк, то результат умножения на 3 необходимо записывать в 16-разрядное слово с тем, чтобы значения смещения могли быть больше 256.

Смещение двоичной запятой

При выполнении арифметических операций над числами, представляющими дробные значения, часто приходится изменять положение двоичной запятой. Так, для операций сложения и вычитания двоичные запятые в обоих операндах должны быть выравнены. При этом в получающемся результате положение двоичной запятой остается таким же, как и в операндах:

$$\begin{array}{r} 00101,001 \\ + 01010,110 \\ \hline 01111,111 \end{array}$$

Если двоичные запятые не выравнены, то в каком-то из операндов необходимо выполнить сдвиг. При выполнении операции сдвига вправо для операнда, у которого двоичная запятая расположена левее, младшие разряды теряются. В принципе можно добиться того, чтобы сдвинутый операнд можно было сохранить в регистре.

$$\begin{array}{r} \text{Первоначальный операнд} \quad 101,00101 \\ \text{Сдвиг вправо} \quad + 00101,001 \\ \quad \quad \quad 01010,110 \\ \hline \quad \quad \quad 01111,111 \end{array}$$

Потерю младших значащих разрядов можно предотвратить за счет сдвига кода операнда влево. Однако при этом необходимо предусмотреть соответствующие меры, чтобы избежать потери старших значащих разрядов и искажения результата.

При перемножении двух двоичных чисел, в которых двоичная запятая отделяет справа соответственно b_1 и b_2 разрядов, в полученном произведении двоичная запятая будет определять справа $(b_1 + b_2)$ разрядов.

$$\begin{array}{r} \times 00101,000 \\ 000011,00 \\ \hline 00000001111,00000 \end{array}$$

Запятая отделяет справа 3 разряда
Запятая отделяет справа 2 разряда
Запятая отделяет справа 5 разрядов

При делении двоичных чисел запятая частного отделяет справа $(b_1 - b_2)$ разрядов, где b_1 и b_2 — число разрядов, отделяемых справа двоичными запятыми в делимом и делителе соответственно.

Операция сдвига для двоичных чисел в дополнительном коде

Как отмечалось ранее, для умножения двоичных чисел без знака на 2 может быть использована операция сдвига влево на один разряд, а для деления на 2 — операция сдвига вправо

на один разряд. То же самое можно сделать и для двоичных чисел со знаком. Для того чтобы двоичное число в дополнительном коде умножить на 2, необходимо выполнить сдвиг кода этого числа влево на один разряд, а в освободившийся младший разряд записать 0. При этом если заранее не известно первоначальное значение числа, то после выполнения сдвига необходимо проверить, не произошло ли изменения знака у результата, для того чтобы выявить возможное переполнение. Заметим, что после операций сдвига влево предыдущее значение знака будет записано в бит переноса регистра флажков.

DOBL:	ADD	A	;СДВИГ ВЛЕВО, ЗАПИСЬ 0 ;В МЛАДШИЙ РАЗРЯД, УСТА- ;НОВКА В 1 ВСЕХ БИТОВ РЕГИСТРА ;ФЛАЖКОВ
	JP	POS	;ПРОВЕРКА НОВОГО ЗНАЧЕНИЯ ;ЗНАКА
	JNC	OFLW	;СТАРОЕ ЗНАЧЕНИЕ ЗНАКА ;БЫЛО РАВНО 0, НОВОЕ—1
	JMP	CONT	;СТАРОЕ ЗНАЧЕНИЕ ЗНАКА ;БЫЛО РАВНО 1, НОВОЕ—1
POS:	JC	OFLW	;СТАРОЕ ЗНАЧЕНИЕ БЫЛО ;РАВНО 1, НОВОЕ—0
CONT:	NOP		;ПЕРЕПОЛНЕНИЕ ОТСУТСТВУЕТ, ;ВЫЧИСЛЕНИЯ ПРОДОЛЖАЮТСЯ

Деление двоичного числа в дополнительном коде на 2 может быть выполнено с помощью операции сдвига вправо на один разряд. При этом, для того чтобы не изменились знак и величина результата, в освобождающийся старший разряд должно быть записано значение знакового разряда, а не просто 0.

HALF:	ORA	A	;УСТАНОВКА В 1 БИТА ЗНАКА, ;ОЧИСТКА БИТА ПЕРЕНОСА
	JP	ROT	;ПОЛОЖИТЕЛЬНОЕ ЧИСЛО, ;ПЕРЕНОС РАВЕН 0
	CMC		;ОТРИЦАТЕЛЬНОЕ ЧИСЛО, ;ПЕРЕНОС РАВЕН 1
ROT:	RAR		;ДЕЛЕНИЕ ЧИСЛА НА 2, УСТА- ;НОВКА ПРАВИЛЬНОГО ЗНАЧЕНИЯ ;ЗНАКА

Числа с плавающей запятой

Числа, о которых шла речь до этого, являлись числами с фиксированной запятой. Для представления чисел, значения которых больше, чем те, которые в предельном случае могут быть выражены заданным числом двоичных разрядов, необходимо слева от двоичной запятой ввести дополнительные разряды. А для представления достаточно малых чисел необходимо использовать дополнительные разряды справа от запятой. Ввиду этого представление чисел с фиксированной запятой представляется несколько громоздким. Кроме того, для исключения ошибок вычислений необходимо заранее знать диапазон изменения значений чисел на всех этапах выполнения алгоритма. Исходные данные для программы обычно поступают с каких-нибудь внешних устройств (терминалов, датчика температуры и др.). Поскольку диапазон входных данных и промежуточных результатов может быть достаточно большим, то для представления таких чисел потребуется большое число разрядов.

Преодолеть указанные трудности позволяет представление чисел в формате с плавающей запятой. Каждому такому числу соответствует пара чисел в формате с фиксированной запятой — мантисса и показатель:

$$(\text{МАНТИССА}) \times 2^{(\text{ПОКАЗАТЕЛЬ})}.$$

Точность представления чисел с плавающей запятой зависит от значения числа. Для представления наибольшего и наименьшего чисел в этом случае потребуется относительно немного разрядов. В микропроцессорах под мантиссу обычно отводится от 8 до 32 разрядов, а под показатель — от 6 до 16 разрядов. Поскольку диапазон значений показателя и мантиссы определяется программистом на этапе составления подпрограмм, реализующих арифметические действия над числами с плавающей запятой, то разрядность мантиссы и показателя можно выбрать, исходя из требований задачи. К сожалению, написание таких программ достаточно трудоемко; поэтому рекомендуется пользоваться стандартными подпрограммами, выполняющими необходимые арифметические действия с числами в формате с плавающей запятой.

Д. Гивоне, Р. Россер

Микрокомпьютер состоит из различных компонент, в число которых в общем случае входят микропроцессор, модули памяти, регистры ввода-вывода, периферийные устройства. Объединение этих компонент составляет существо задачи проектирования микрокомпьютерных систем и должно производиться с учетом характера и временных параметров сигналов на стыках между компонентами. Чтобы сигналы были совместимыми, вообще говоря, нужно правильно выбрать компоненты и спроектировать специальные вспомогательные схемы, называемые **интерфейсами**.

В отношении микрокомпьютеров можно выделить две основные проблемы интерфейса (проблемы стыковки). Одна проблема состоит в подключении таких компонент, как модули памяти и регистры ввода-вывода, к шинам микропроцессора. Здесь главное — это синхронизация и управление шинами, а также выборка компонент, обеспечивающая своевременную передачу данных между микропроцессором и выбранной компонентой.

Другая проблема касается стыковки частей микрокомпьютера с внешними компонентами, такими, как периферийные устройства, каналы передачи данных, контроллеры и т. п. Этот интерфейс не выходит непосредственно на шины микрокомпьютера, и поэтому он существенно меньше структуризован. К нему относится преобразование внешних сигналов, которые могут иметь любую природу (в том числе и аналоговую), в сигналы, совместимые с сигналами на шинах, и обратное преобразование.

27.1. ПОРТЫ ВВОДА-ВЫВОДА

В соответствии с общепринятым соглашением направление потоков вводной и выводной информации рассматривается от-

¹⁾ Adapted from *Microprocessors/Microcomputers: An Introduction*, by Donald D. Givone and Robert R. Roesser. Copyright © 1980. Used by permission of McGraw-Hill, Inc. All rights reserved. [Имеется перевод: Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры. — М.: Мир, 1983, с. 327].

носителем микропроцессора. Поэтому **портом ввода** называется любой источник данных, например адресуемый регистр, подключенный к шине данных микрокомпьютера. Он выдает слово в микропроцессор, когда к нему происходит обращение. **Портом вывода** называется приемник данных, например адресуемый регистр, подключенный к шине данных микропроцессора. Он получает слово от микропроцессора, когда последний к нему обращается. В большинстве микропроцессоров для адресации портов (т. е. для выборки нужного порта) используется адресная шина или ее часть. Довольно часто адреса портов ввода отличаются от адресов портов вывода и от адресов памяти не значениями, а сигналами на соответствующих управляющих линиях.

Порты ввода-вывода иллюстративного микропроцессора

На рис. 27.1 показана типичная схема порта ввода и порта вывода для нашего микропроцессора. Этим двум портам присвоен уникальный 8-битовый код (адрес устройства), по которому микропроцессор может отличить их от других портов. В нашем случае обоим портам присвоен один код (01100101), что вполне допустимо, поскольку между собой они будут отличаться сигналами в управляющих линиях «ввод» и «вывод». При выполнении команд ввода или вывода на младшую половину адресной шины подается код устройства, взятый из второго байта команды. Специальный вентиль И на восемь входов служит для распознавания кода данного устройства. Входы вентиля подключены к 8 младшим линиям адресной шины, причем в зависимости от кода часть сигналов инвертируется. Выход вентиля И («выборка») используется для выборки обоих портов.

Порт ввода имеет 8-битовый регистр, на который поступает информация от некоторого внешнего устройства. Выходы этого регистра подключены к шине данных через восемь тристабильных формирователей. Сигнал разрешения на эти формирователи есть результат логического И сигнала «выборка» и управляющего сигнала «ввод» от микропроцессора.

Содержимое регистра подается на шину данных, когда (1) «выборка» равна логической 1, т. е. задан код устройства, относящийся к данному порту, и (2) «ввод» равен логической 1, т. е. выполняется команда ввода, и в ее фазе выполнения наступил момент ввода данных. Задача порта — только поместить информацию на шину данных. Дальнейший ее путь к месту назначения (к аккумулятору в данном случае) определяется микропроцессором.

Передача данных осуществляется в третьем машинном цикле команды ввода. Временные диаграммы сигналов в этом цикле

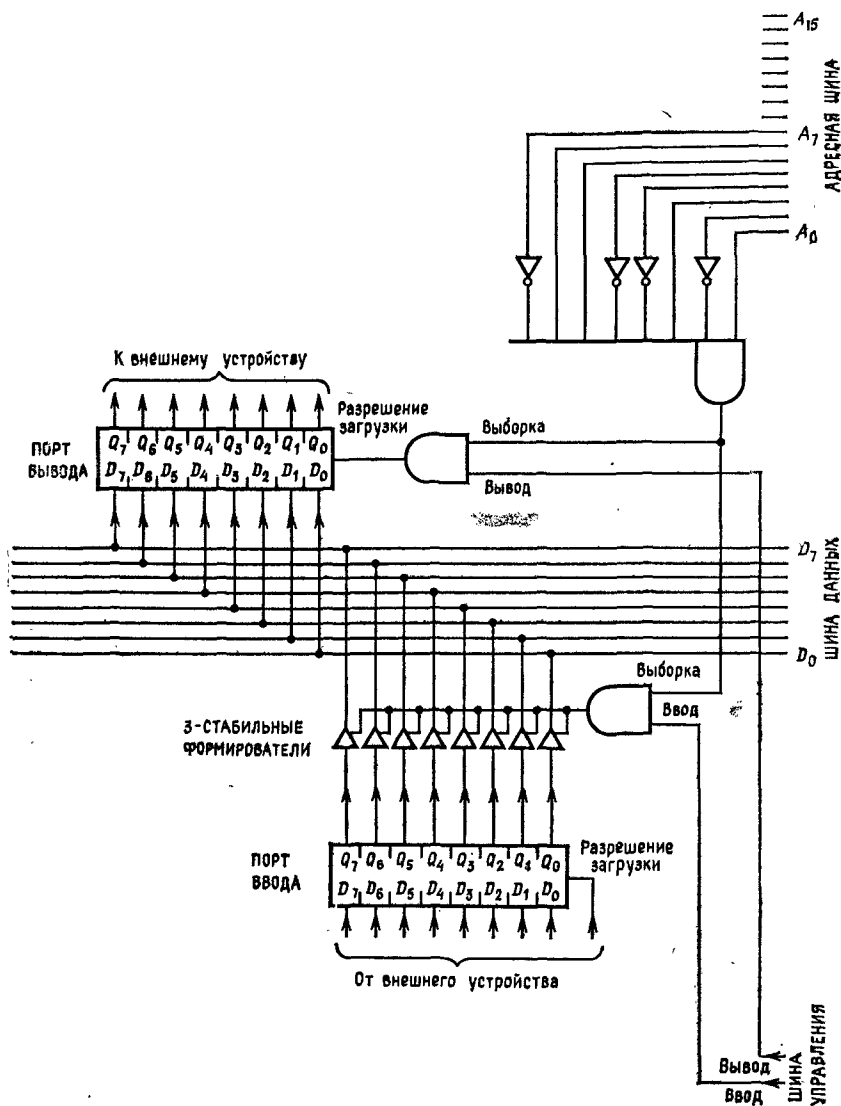


Рис. 27.1. Схема порта ввода и порта вывода в иллюстративном микропроцессоре.

показаны на рис. 27.2. В машинный цикл входит три синхронимпульса. В начале первого синхронимпульса T_1 микропроцессор подает код соответствующего порта на младшую половину адресной шины. По концу T_1 подается логическая 1 на линию

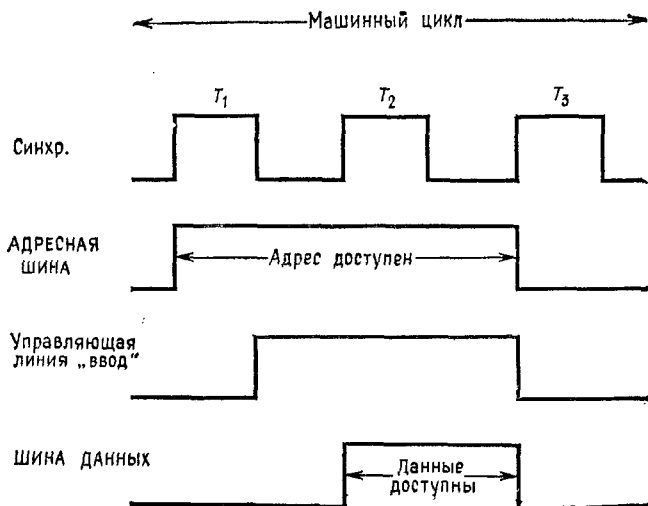


Рис. 27.2. Временные диаграммы ввода данных.

«ввод», которая остается в этом состоянии до переднего фронта импульса T_3 . При этом предполагается, что на шину данных в ответ на сигнал «ввод» будут поданы данные с регистра порта ввода, и они будут доступны в интервале между передними фронтами T_2 и T_3 . Стробирование передачи данных на аккумулятор осуществляется по концу T_2 .

Нужно отметить, что описанная логика работы порта ввода налагает два ограничения на время срабатывания схем. Одно ограничение связано с тем, что информация должна быть подана на шину данных не позднее, чем через полтакта после прихода сигнала «ввод». Это означает, что задержка в триггерах формирования плюс задержка в двухходовом вентиле И, управляющем усилителями, не должна превышать одного такта.

В конкретных микропроцессорах мы можем встретить схемы ввода данных, отличающиеся несколько от описанной, однако аналогичные временные ограничения существуют и для них. Как правило, этим требованиям во внешних компонентах удовлетворить легко, поскольку элементы, из которых строятся порты ввода-вывода, обычно оказываются более быстродействующими по сравнению с элементами, из которых строится микропроцессор. Например, в портах ввода-вывода можно использовать ТТЛ-схемы средней степени интеграции, более быстрые по сравнению с большими интегральными МОП-схемами, применяемыми в микропроцессоре,

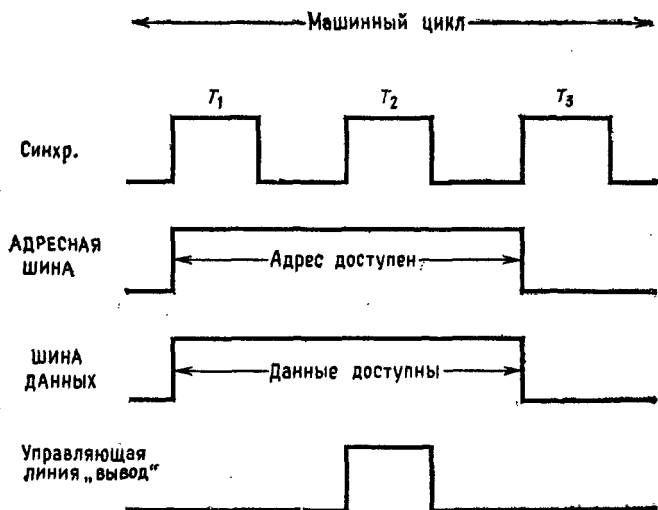


Рис. 27.3. Временные диаграммы вывода данных.

Порт вывода, показанный на рис. 27.1, имеет 8-битовый регистр, используемый для передачи информации внешнему устройству. Регистр построен из синхронных фиксаторов D -типа, управляемых общей линией разрешения загрузки. Входы D -триггеров подсоединены непосредственно к шине данных. Сигнал «разрешение загрузки» формируется на двухходовом вентиле И из сигналов «выборка» и «вывод». Таким образом, информация на шине данных, поступившая из аккумулятора микропроцессора, передается в регистр порта вывода, когда (1) «выборка» равна логической 1, т. е. задан код данного порта, и (2) «вывод» равен логической 1, т. е. выполняется команда вывода, и в ее фазе выполнения наступил момент вывода данных.

Передача данных осуществляется в третьем машинном цикле команды вывода. Временные диаграммы сигналов в этом цикле показаны на рис. 27.3. По переднему фронту синхроимпульса T_1 микропроцессор подает код устройства на младшую половину адресной шины и данные на шину данных. Затем в момент T_2 выдается сигнал «вывод», стробирующий передачу данных на регистр вывода.

Схемы порта вывода, так же как и порта ввода, должны удовлетворять определенным временным ограничениям, и так же, как и для портов ввода, эти ограничения обычно нетрудно соблюсти благодаря относительно высокому быстродействию схем ввода-вывода.

Варианты схем ввода-вывода

Различия между только что рассмотренной логикой работы ввода-вывода в иллюстративном микропроцессоре и логикой работы в некоторых промышленных микропроцессорах в основном касаются временных соотношений и смысла управляющих сигналов. В нашем микропроцессоре сигналы «ввод» и «вывод» выполняли и управляющие, и стробирующие функции. Во многих промышленных микропроцессорах сигналы, аналогичные сигналам «ввод» и «вывод», получаются как комбинация других управляющих сигналов.

В некоторых случаях предусматривается управляющая линия, которая задает направление передачи по шине данных. Эта линия, назовем ее, скажем, «вывод данных», при значении 1 задает вывод, а при значении 0 — ввод. Для задания момента передачи служит либо синхролиния, либо другая управляющая линия. Интерфейс портов ввода-вывода с таким микропроцессором можно реализовать аналогично интерфейсу с иллюстративным микропроцессором, сформировав сигналы «ввод» и «вывод» следующим образом:

$$\text{Ввод} = (\text{Вывод данных}) \text{ Строб},$$
$$\text{Вывод} = (\text{Вывод данных}) \text{ Строб},$$

где «строб» — это сигнал, определяющий момент передачи.

В некоторых схемах в начале машинного цикла по шине данных посылается управляющее слово, которое определяет тип предстоящей передачи: ввод или вывод, чтение из памяти или запись в память. Это управляющее слово запоминается во внешнем регистре и в комбинации с синхросигналами или с другими управляющими сигналами используется для образования сигналов, аналогичных сигналам «ввод», «вывод»; «чтение» и «запись» нашего иллюстративного микропроцессора. При таком подходе некоторые управляющие линии становятся ненужными.

27.2. КООРДИНАЦИЯ ВЗАИМОДЕЙСТВИЯ С ВНЕШНИМИ УСТРОЙСТВАМИ

Как отмечалось в предыдущем разделе, порты ввода-вывода — это средство подключения внешних устройств к микропроцессору. В качестве внешних устройств могут выступать такие приемники и источники цифровой информации, как внешние запоминающие устройства, терминалы, измерительные приборы, контроллеры различных станков или даже другие микропроцессоры.

Большинство внешних устройств работает асинхронно по отношению к микропроцессору. Например, терминал, состоящий из клавиатуры и печатающего устройства, порождает данные при каждом нажатии кнопки человеком и воспринимает данные со скоростью, определяемой конструкцией печатающего механизма. Поэтому возникает проблема согласования моментов срабатывания внешнего устройства и микропроцессора. В противном случае при вводе передача может произойти в тот момент, когда данные еще не готовы, а при выводе — когда предыдущие данные еще не восприняты устройством. Следовательно, взаимодействие между внешним устройством и микропроцессором должно происходить по определенным правилам, которые иногда называют **протоколом взаимодействия**, или **рукопожатием**.

Вообще говоря, работу внешнего устройства можно разбить на последовательность некоторых действий, определяемых его назначением и конструкцией. В результате каждого такого действия либо воспринимается одно слово, находящееся в порту вывода, либо выдается одно слово в порт ввода. Например, действие клавиатуры терминала заключается в нажатии на одну из клавиш и передачи соответствующего кода в порт ввода. Аналогичным образом действие печатающего механизма сводится к подаче тока в соленоид молоточка того символа, код которого находится в порту вывода.

Вообще говоря, предполагается, что порт во время выполнения действия в устройстве занят устройством, и, следовательно, микропроцессор не должен к нему обращаться. И только после завершения действия порт становится доступным для процессора, т. е. готовым к передаче данным. Процесс передачи данных между процессором и устройством ввода-вывода состоит из следующих трех операций:

1. Запуск (инициация) действия в устройстве.
2. Проверка микропроцессором готовности порта.
3. Собственно передача данных.

Методы организации ввода-вывода обычно различаются в соответствии с теми способами, которыми реализуются первые две операции.

Пуск со стороны программы

Один из методов организации ввода-вывода характеризуется программным запуском действия в устройстве для каждого акта передачи данных. При этом сигналы запуска посылаются от процессора к устройству в моменты времени, определяемые программой. Устройство реагирует на эти сигналы выдачей очередного слова в порт ввода или приемом слова из порта вы-

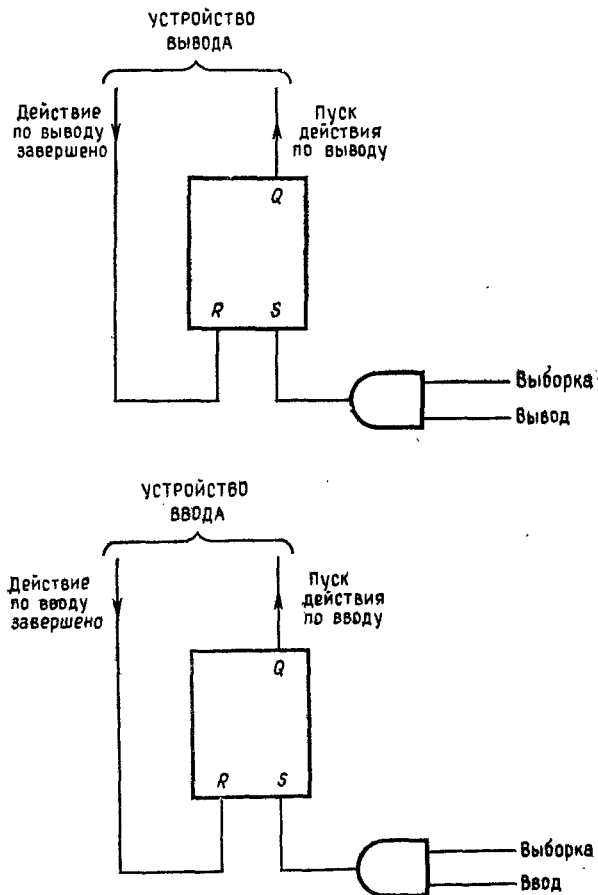


Рис. 27.4. Управляющие триггеры для пуска действия в устройстве ввода-вывода.

вода. После того как пройдет время, достаточное для завершения действия в устройстве, программа может снова обращаться к порту. Очень часто само обращение к порту служит сигналом, запускающим действие в устройстве. Например, для случая ввода в результате этого действия в порту появятся данные, которые будут пересылаться при следующем обращении микропроцессора к порту.

В качестве иллюстрации рассмотрим считыватель и перфоратор бумажной ленты, подключенные соответственно к порту ввода и порту вывода. Будем считать, что на перфоленте

последовательно расположены 8-битовые слова. Предположим, что первоначально порт ввода уже заполнен первым словом, полученным считывателем с перфоленты. Обращения к порту ввода могут происходить в разных точках программы.

Помимо того что данные выбираются из порта ввода, каждое обращение служит сигналом для запуска считывателя. Считыватель в ответ на этот сигнал читает следующее слово с перфоленты и замещает им прежнее слово в порте ввода. В других точках работающей программы могут выполняться обращения к порту вывода. Помимо загрузки данных в регистр порта, эти обращения служат сигналом для запуска перфоратора. В ответ на этот сигнал он перфорирует на ленте слово, находящееся в регистре порта.

На рис. 27.4 показано, как сигналы пуска могут быть сформированы иллюстративным микропроцессором. Схемы приведены для устройств как ввода, так и вывода. В каждой схеме используется асинхронный RS-триггер, управляющий внешним устройством. Установка триггеров осуществляется теми же сигналами, которые использовались на рис. 27.1 для управления передачей данных между микропроцессором и соответствующим регистром порта. В частности, триггер выводного порта устанавливается логическим И от сигналов «выборка» и «вывод», а триггер вводного — логическим И от сигналов «выборка» и «ввод».

Выходной сигнал от каждого триггера поступает в устройство и вызывает в нем выполнение действия, например протяжку перфоленты и возбуждение перфорирующих соленоидов. Результатом действия, во всяком случае, должно быть принятие выводимых или поставка вводимых данных. После завершения действия устройство подает сигнал сброса соответствующего управляющего триггера. Этим обеспечивается однократное выполнение действия в устройстве.

Пуск со стороны устройства

Возможен другой метод организации ввода-вывода, при котором действие в устройстве по поводу каждого акта обмена данными запускается (инициируется) самим устройством. В этом случае программа должна получать информацию о завершении действия в устройстве и о готовности порта к очередной передаче данных. Для этого служит флаг готовности порта, который устанавливается устройством и периодически опрашивается программой, ожидающей обмена с данным портом. Обычно для опроса состояния одного или группы устройств ввода-вывода используется вспомогательный порт ввода — порт состояний. Разряды, входящие в этот порт, характеризуют те

или иные условия, относящиеся к группе устройств, и в частности состояния готовности их портов.

В программе, осуществляющей ввод-вывод через некоторый порт, сначала спрашивается его готовность, для чего вводится содержимое соответствующего порта состояния. Бит готовности выделяется и тестируется. Если выясняется, что порт не готов, то операция опроса готовности повторяется.

Типичным представителем устройств, в которых запуск действия осуществляется по инициативе самого устройства, можно считать магнитную ленту. Данные на ленте обычно объединяются в блоки, называемые записями и состоящие из многих слов. Между записями оставляются пустые промежутки. Промежутки нужны для торможения ленты и последующего разгона до постоянной рабочей скорости. Поэтому в процессе чтения или записи слова данных следуют друг за другом с постоянной скоростью. Обработка одного слова и составляет действие устройства.

Более общая схема обмена

Объединив оба рассмотренных выше метода, можно получить более общую схему обмена. В этом случае программа инициатирует действие в устройстве, и проверяет готовность порта.

На рис. 27.5 показано, как можно в нашем иллюстративном микропроцессоре реализовать интерфейс по такой общей схеме как для вводного, так и выводного порта. Если устройство способно и вводить, и выводить данные, оно подключается сразу к двум портам. Помимо этих двух портов для данных, на рисунке показаны порт состояния и порт управления.

Порт состояния — это порт ввода, через который микропроцессор может получить наиболее важную информацию о состоянии устройства. В частности, неготовность вводного порта данных определяется битом 7, а неготовность выводного порта данных — битом 0. Оба этих бита поступают с выходов асинхронных RS-триггеров. В каждом из этих триггеров устанавливается 1 на время выполнения передачи данных между микропроцессором и портом, т. е. на время, когда порт занят. Это же состояние сообщает устройству о необходимости выполнить соответствующее действие, т. е. выдать новые данные в случае порта ввода или принять данные в случае порта вывода. Завершив действие, устройство сбрасывает соответствующий бит состояния, и это говорит микропроцессору о готовности порта. Остальные шесть битов состояния могут характеризовать другие возможные ситуации в устройстве, такие, как «конец ленты», «конец блока», «ошибка при вводе» и т. п.

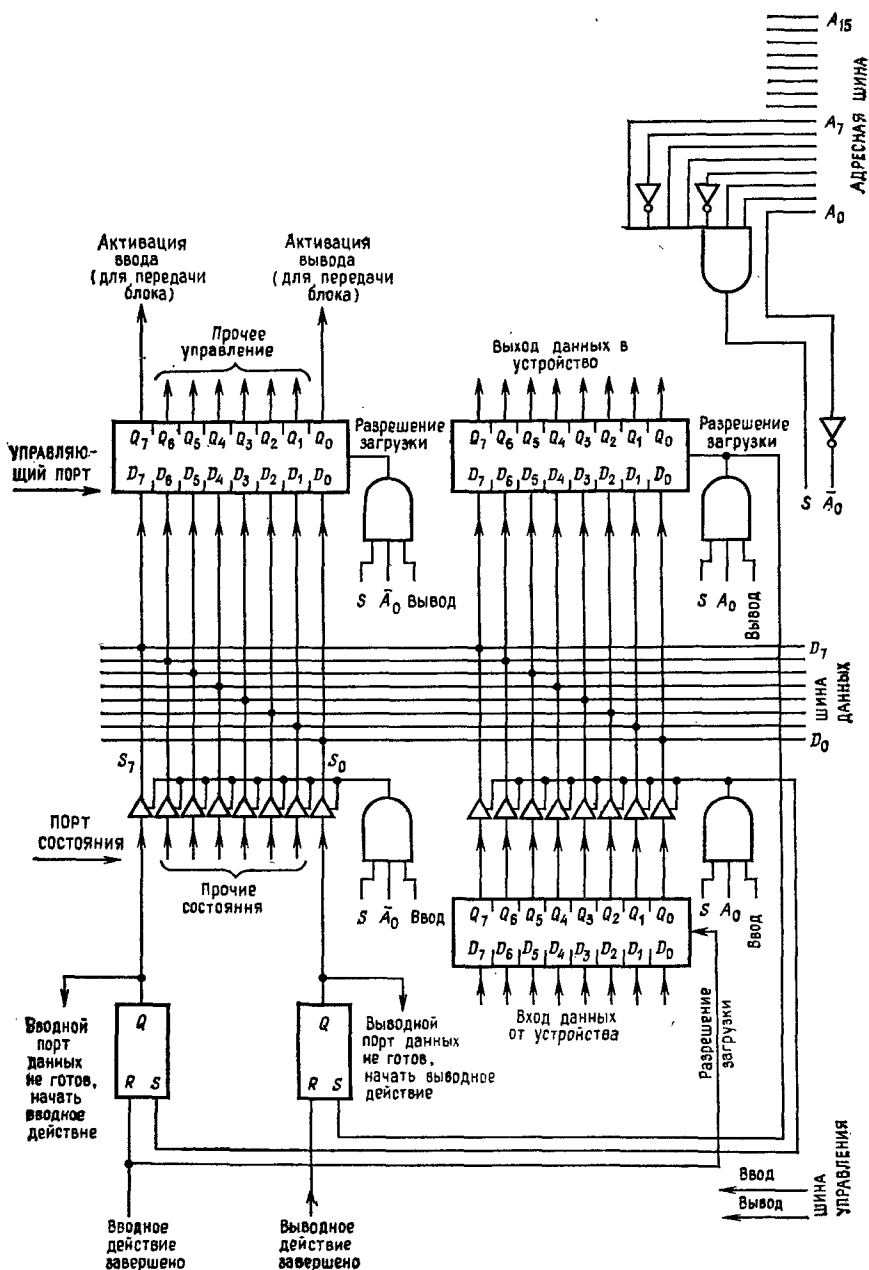


Рис. 27.5. Общий интерфейс иллюстративного микропроцессора с устройством ввода-вывода, включающий порты ввода и вывода данных, порт состояния и порт управления.

Порт управления — это выводной порт, который служит для передачи команд устройству от микропроцессора. Две из этих команд соответствуют битам 7 и 0 и называются «активация ввода» и «активация вывода». Назначение этих команд — чтение или запись целого блока данных. Например, при чтении магнитной ленты бит «активация ввода» устанавливается в 1 в начале блока данных и сбрасывается в 0 после того, как под магнитными головками пройдут все слова блока данных.

В устройстве значения этих командных битов объединяются вентилем И с битами неготовности портов для того, чтобы начать или продолжить выполнение действий в устройстве, связанных с актами передачи данных. Так, например, движение ленты и чтение информации с нее продолжается только, если оба бита «активация ввода» и «неготовность порта» имеют значение 1. Оставшиеся шесть управляющих битов можно использовать для выполнения таких функций, как «перемотка ленты», «пропуск блока», «сброс флагов ошибок» и т. п.

Для выборки рассмотренных четырех портов предусмотрена специальная схема. Она включает вентиль И с семью входами, подключенными к линиям младшей половины адресной шины, кроме линии A_0 . Выход S этого вентиля принимает значение логической 1, если на адресную шину подан любой из двух кодов 10110110 или 10110111. Первый из них соответствует портам состояния и управления, а второй — портам данных.

Каждый порт управляется своим трехходовым вентилем И. Сигнал S подается на один из входов в каждом вентиле. Инвертированное значение \bar{A}_0 младшего разряда адресной шины подается на входы вентилях двух портов, а именно портов состояния и управления, тогда как на вентили портов данных подается сигнал A_0 без инверсии. Таким образом, каждой паре портов присваивается некоторый код устройства. На третий вход вентилях И подается сигнал «ввод», если порт вводной, и сигнал «вывод», если порт выводной. Таким образом, каждый порт может быть однозначно указан микропроцессором. Кроме того, выходы двух из рассмотренных управляющих вентилях поданы на входы установки соответствующих триггеров состояния.

В табл. 27.1 приведены программы ввода и вывода, которые могут работать со схемой рис. 27.5. Программа ввода начинается в ячейке 1000₁₆. Первая команда вводит слово состояния устройства из порта ввода В6₁₆. Бит неготовности вводного порта данных (бит 7) выделяется из слова состояния при помощи логического умножения на маску 80₁₆. Если этот бит равен 0, что говорит о наличии в порте данных для ввода, то в аккумуляторе будет 0. При невыполнении этого условия следующая команда вызовет переход на повторение проверки

Таблица 27.1. Фрагменты программы, осуществляющие ввод и вывод

Ячейка памяти	Команда на машинном языке	Команда в символической форме	Комментарий
1000	FD	INP	Ввод слова состояния
1001	B6	B6	
1002	61	LRI 1	Загрузка маски в H
1003	80	80	
1004	81	AND 1	Выделение бита 7
1005	7D	JAN	Тест бита 7; если равен 1, повторить проверку
1006	10	10	
1007	00	00	
1008	FD	INP	Ввод данных
1009	B7	B7	
2000	FD	INP	Ввод слова состояния
2001	B6	B6	
2002	61	LRI 1	Загрузка маски в H
2003	01	01	
2004	81	AND 1	Выделение бита 0
2005	7D	JAN	Тест бита 0; если равен 1, повторить проверку
2006	20	20	
2007	00	00	
2008	FE	OUT	Вывод данных
2009	B7	B7	

состояния. В противном случае следующая команда осуществит ввод данных из порта B7, чем завершается фрагмент программы.

Программа вывода начинается в ячейке 2000₁₆. Она аналогична программе ввода с той разницей, что в качестве бита готовности порта анализируется бит 0. Он выделяется при помощи маски 01₁₆. Кроме того, последняя команда, естественно, является командой ввода.

27.3. ПРЕРЫВАНИЯ ПРОГРАММЫ

Важнейшее значение имеет способность большинства микропроцессоров прервать выполняемую программу в ответ на внешнее событие и выполнять программу, специально предназначенную для обработки этого события. Это называется **прерыванием программы**. Например, прерывающим событием может оказаться завершение ожидаемого микропроцессором действия во внешнем устройстве. Воспользоваться прерыванием по готовности порта, как правило, много удобнее и эффективнее, чем не-

прерывно опрашивать его состояния. Микропроцессор освобождается для выполнения других функций, и, в частности, может заняться другим портом ввода-вывода.

Прерывание программы для большинства компьютеров напоминает переход на подпрограмму с той разницей, что оно инициируется не командой в программе, а приходом внешнего сигнала по управляющей линии. Этот сигнал называется **запросом на прерывание**. Так же как и подпрограмма, программа обработки прерывания размещается в памяти, начиная с ячейки, в которую должно передаваться управление.

Обнаружив запрос на прерывание, процессор откладывает выполнение текущей программы и начинает выполнять программу прерывания. Программа прерывания обычно заканчивается командой возврата, после которой продолжается выполнение прерванной программы. Обычно процессор обладает возможностью запрещать (блокировать) прерывания на какие-то отрезки времени, когда их обработка по тем или иным причинам неудобна. При заблокированных прерываниях поступающие запросы на прерывания игнорируются.

Обычно в микропроцессорных системах запросы на прерывания могут поступать от нескольких устройств. Поэтому возникает проблема идентификации устройства, приславшего запрос, с тем чтобы можно было выполнить действия по обслуживанию именно этого устройства. Существуют два основных метода решения этой проблемы. Согласно одному из них, должна существовать главная программа обработки прерываний, которая при поступлении запроса проверяет состояние каждого устройства и находит устройство, требующее обслуживания. Такую схему часто называют системой **прерываний с программным опросом**. При другом методе информацию, которая идентифицирует устройство, приславшее запрос, формирует аппаратура. Такую систему прерываний иногда называют **векторной приоритетной системой**.

Прерывания с программным опросом

В схеме прерываний с программным опросом все запросы на прерывания поступают по одной управляющей линии. Эта линия является выходом вентиля ИЛИ, на входы которого поступают запросы от индивидуальных устройств. Каждому устройству обычно выделяется порт состояния, а в нем отводится один бит, хранящий запрос на прерывание. Когда по общей линии в микропроцессор поступает запрос от любого устройства, выполнение текущей команды завершается и, если прерывания не заблокированы, происходит передача управления в ячейку с фиксированным адресом. С этой ячейки начинается **главная**

программа обработки прерываний, которая последовательно вводит содержимое портов состояний и тестирует биты запросов на прерывания. Обнаружив устройство, запросившее обслуживание, главная программа передает управление программе, обслуживающей данное устройство.

Когда запросы поступают одновременно от двух или большего числа устройств, возникают конфликтные ситуации. Их разрешение заложено в самой схеме опроса. Первым обслуживается то устройство, которое раньше попадает в порядке опроса. Конечно, после обслуживания одного запроса будет обслужен следующий из отложенных запросов и т. д. Таким образом, порядок опроса определяет приоритетность обслуживания устройств.

Главный недостаток такой системы прерываний связан со временем, затрачиваемым программой на опрос состояния отдельных устройств. Часто эта задержка не играет существенной роли, поскольку во многих приложениях микропроцессоры в отличие от больших машин работают с немногочисленными внешними устройствами. Тем не менее существуют приложения, более чувствительные к подобным задержкам. Для них предпочтительнее векторная система прерываний, поскольку в ней непосредственно указывается устройство, выдавшее запрос.

Векторная система прерываний

В векторной системе прерываний устройство, вызвавшее прерывание, идентифицируется с помощью внешних по отношению к микропроцессору схем. Конечно, эти схемы должны разрешать конфликты, возникающие при одновременном поступлении нескольких запросов, и предоставлять обслуживание только одному устройству.

Существуют различные способы идентификации прерывающих устройств. Можно, например, иметь в микропроцессоре не одну, а большее число линий для запросов на прерывания и каждую из них предоставить отдельному устройству. В этом случае сигнал на каждой конкретной линии должен вызывать передачу управления в свою ячейку памяти, если только эта передача внутри микропроцессора не заблокирована сигналом прерывания с более высоким приоритетом.

Ячейки, в которые передается управление, являются начальными ячейками программ, обслуживающих различные устройства. Этот способ широко применяется в больших универсальных ЭВМ, но в микропроцессорах встречается редко. При большом числе устройств требуется много линий, а количество выводов микропроцессора существенно ограничено.

Векторная система прерываний с идентификацией устройств при помощи адресов

Способ, которым можно прямо или косвенно воспользоваться в любом микропроцессоре, заключается в том, чтобы посылать в микропроцессор начальный адрес обслуживающей прерывание программы по адресной шине. Этот адрес формируется внешней схемой, которая выбирает устройство для обслуживания в соответствии с некоторой приоритетной схемой. Конечно, если запрос на прерывание поступил только от одного устройства, то именно оно и выбирается.

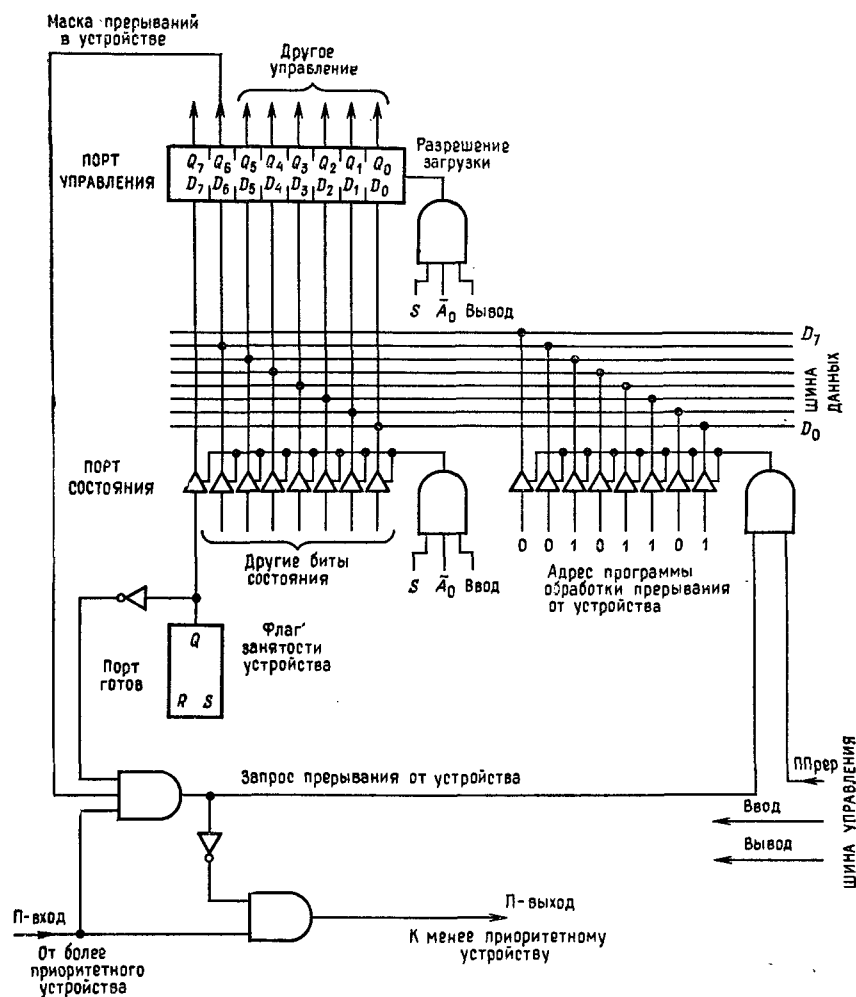
Иллюстративный микропроцессор был задуман в расчете на такую схему, и сейчас мы ее опишем.

Для обработки прерываний в микропроцессоре предусмотрены две управляющие линии: «Прер» (прерывание) и «ППер» (подтверждение прерывания). Линия «Прер», входная для микропроцессора, служит для передачи внешних запросов на прерывание (состояние 1 соответствует наличию запроса). Однако микропроцессор реагирует на запросы прерываний только, если специальный внутренний триггер, флаг прерываний, установлен в 1. Линия «ППер», выходная для микропроцессора, служит для стробирования подаваемого извне на адресную шину адреса перехода на программу прерывания.

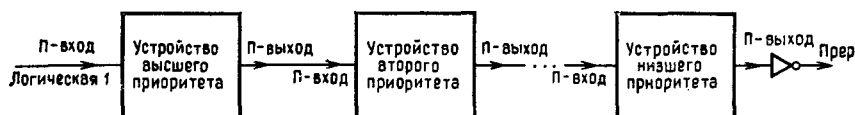
На рис. 27.6, а показаны схемы для одного устройства. Предполагается, что устройству принадлежит один порт для ввода данных (на рисунке не показан) и, кроме того, вводный порт состояния и выводной порт управления. Один бит в порте состояния, крайний левый, играет особую роль в рассматриваемом случае. Этот бит, так же как и на рис. 27.5, является сигналом устройству начать операцию ввода. Этот же бит сообщает микропроцессору о занятости устройства и неготовности порта данных. Инверсия этого бита, а именно сигнал «порт готов», используется для формирования запроса на прерывание от данного устройства. Тем не менее, для того чтобы устройство послало запрос на прерывание, должны быть выполнены еще два условия.

Одно из этих условий требует, чтобы определенный бит в порте управления, называемый **маской прерываний**, был равен 1. С его помощью программа при необходимости может заблокировать прерывания от данного конкретного устройства, не запрещая прерываний от других устройств.

И наконец, второе и последнее условие состоит в том, чтобы другие, более приоритетные устройства дали разрешение данному устройству. С этой целью строится цепочка, как показано на рис. 27.6, б, которая связывает устройства друг с другом в порядке присвоенных им приоритетов. Каждое устройство в



а



б

Рис. 27.6. Векторная система прерываний с адресами прерываний, приходящими от устройств, и с использованием цепочки приоритетов.

а — схема выдачи адреса прерывания от устройства; б — связь в цепочке приоритетов.

цепочке приоритетов имеет входную линию «П-вход» и выходную линию «П-выход».

На линии «П-вход» поддерживается логическая 1, когда ни одно из устройств с более высоким приоритетом не запрашивает прерывания. Эта линия соединена с линией «П-выход» ближайшего более высокого по приоритету устройства. В свою очередь данное устройство формирует сигнал на своей линии «П-выход», который подается на линию «П-вход» в устройство с ближайшим более низким приоритетом. Сигнал «П-выход» данного устройства формируется как логическое И от сигнала «П-вход» и инверсии запроса на прерывание от данного устройства.

Таким образом, приоритет предоставляется устройству, если устройство с более высоким приоритетом не запрашивает прерывания. Три сигнала «маска прерывания», «порт готов» и «П-вход», соответствующие трем упоминавшимся выше условиям, подаются на клапан И и порождают сигнал запроса прерывания от данного устройства.

Когда хотя бы в одном устройстве сигнал запроса прерывания принимает значение 1, такое же значение подается на управляющую линию «Прер» в микропроцессор. Для этого линия «Прер» через инвертор подключается к последней линии «П-выход». На этой последней линии «П-выход» будет логический 0, если хотя бы одно устройство дало запрос на прерывание.

Получив запрос на прерывание, микропроцессор в соответствующий момент откликнется сигналом подтверждения прерывания «ПППрер». Сигнал «ПППрер» в каждом устройстве поступает на клапан И вместе с запросом прерывания от устройства. Выход клапана используется для возбуждения группы триггеров формирователей, которые подают на адресную шину 8-битовый, соответствующий данному устройству адрес. Таким образом, этот адрес поступает в микропроцессор и задает начальную ячейку программы, обслуживающей прерывание. Передача адреса в микропроцессор занимает один машинный цикл и аналогична передаче данных из порта ввода. Разница только в том, что момент передачи определяется не линией «ввод», а линией «ПППрер».

Внутри микропроцессора события при прерываниях развиваются следующим образом. Если внутренний флаг прерываний сброшен в 0, то запрос на прерывание игнорируется. Если же флаг установлен в 1, то логическая 1 на линии «Прер» будет воспринята как запрос прерывания. После этого сначала завершается выполнение текущей команды. Затем содержимое программного счетчика, т. е. адрес следующей команды, заносится в стек. Флаг прерываний сбрасывается, и тем самым блокируются

все прерывания вплоть до момента, когда флаг будет снова установлен в 1 по команде программы. Наконец, один машинный цикл затрачивается на прием 8-битового адреса от прерывающего устройства. Этот адрес помещается в младшую половину программного счетчика, старшая половина которого сбрасывается. Следовательно, переход осуществляется по адресу, не выходящему за пределы первых 256_{10} ячеек памяти.

Первая команда программы обработки прерывания часто оказывается командой перехода на программу, хранящуюся вне первых 256_{10} ячеек. Таким путем можно освободить место для информации, относящейся к другим прерываниям. Программа обработки прерываний пишется с учетом специфики обслуживаемого устройства. Напомним, что при входе в программу прерываний флаг прерываний автоматически сбрасывается и прерывания блокируются. Программа в процессе работы может установить его в 1, т. е. снять блокировку прерываний, чтобы разрешить другим устройствам, особенно более приоритетным, прерывать данную программу.

Если блокировка прерываний не снимается в середине программы, то это делается в конце предпоследней командой. Блокировка прерываний снимается с задержкой на одну команду, что позволяет до следующего прерывания выполнить последнюю команду программы данного прерывания. Эта последняя команда — всегда команда возврата, она поднимает стек и восстанавливает программный счетчик к тому состоянию, которое он имел в момент прерывания.

Векторная система прерываний с шифратором приоритетов

В другом варианте векторной системы прерываний, который можно применять с любым микропроцессором, процессору сообщается номер прерывающего устройства. В такой системе прерывание вызывает переход на фиксированную ячейку, где начинается главная программа обработки прерываний. Эта программа вводит данные из специального порта, содержащего номер устройства, запросившего прерывания. Зная этот номер, главная программа передает управление программе, обслуживающей данное устройство. Как и в других схемах, конфликты при одновременных запросах на прерывания от нескольких устройств разрешаются в соответствии с приоритетами устройств.

Такую схему можно реализовать аналогично предыдущей схеме, где приоритеты определялись приоритетной цепочкой, а код номера устройства указывался самим прерывающим устройством. Однако часто можно встретить другую реализацию этих двух функций. В ней код номера устройства с наибольшим

приоритетом, выдавшего запрос на прерывание, формируется комбинационной схемой, называемой **шифратором приоритетов**. На входы шифратора подаются сигналы запросов на прерывание от отдельных устройств. Входы перенумерованы, начиная с 0, и номер входа соответствует номеру подключенного к этому входу устройства. На выходных линиях шифратора формируется двоичный номер старшей среди возбужденных (т. е. содержащих логическую 1) входных линий.

В качестве примера рассмотрим конструкцию шифратора приоритетов, имеющего восемь входов I_0, I_1, \dots, I_7 . Предположим, что большие номера соответствуют более высоким приоритетам. При восьми входах для представления двоичного номера выбранного входа достаточно трех выходов. Обозначим эти выходы через b_2, b_1, b_0 в соответствии со старшинством. Воспользуемся интуитивными соображениями и найдем логические выражения для выходных сигналов. Прежде всего заметим, что b_2 должно равняться логической 1, когда возбужден любой вход с номером, большим или равным 4:

$$b_2 = I_7 + I_6 + I_5 + I_4.$$

Далее заметим, что b_1 должно иметь значение 1, если возбуждены входы I_7 или I_6 , поскольку в двоичном представлении чисел 6 и 7 бит b_1 равен 1. Бит b_1 также равен 1, если возбуждены I_2 или I_3 , но при этом не возбуждены ни I_4 , ни I_5 , поскольку бит b_1 равен 1 в двоичном представлении чисел 2 и 3, но не чисел 4 и 5. Объединение этих двух условий дает выражение для

$$b_1 = I_7 + I_6 + \bar{I}_5 \bar{I}_4 (I_3 + I_2).$$

Наконец, заметим, что b_0 должно иметь значение 1, если возбужден нечетный вход, при условии, что не возбужден ни один из более старших четных входов. Это соображение дает нам следующее выражение для b_0 :

$$b_0 = I_7 + \bar{I}_6 I_5 + \bar{I}_6 \bar{I}_4 I_3 + \bar{I}_6 \bar{I}_2 I_1.$$

Например, рассмотрим случай, когда возбуждены входы I_1, I_2, I_4 и I_5 . Подставляя единичные значения для этих переменных в выражения, мы получим $b_2 = 1, b_1 = 0$ и $b_0 = 1$, что соответствует двоичному представлению числа 5, являющегося номером возбужденного входа с высшим приоритетом.

На рис. 27.7 приведена схема векторной системы прерываний с шифратором приоритетов, которую можно применить в иллюстративном микропроцессоре. Запросы на прерывания от устройств (их число не должно превышать 8) поступают на входы шифратора приоритетов, который определяет номер старшего запроса. Полученный 3-битовый номер поступает в порт

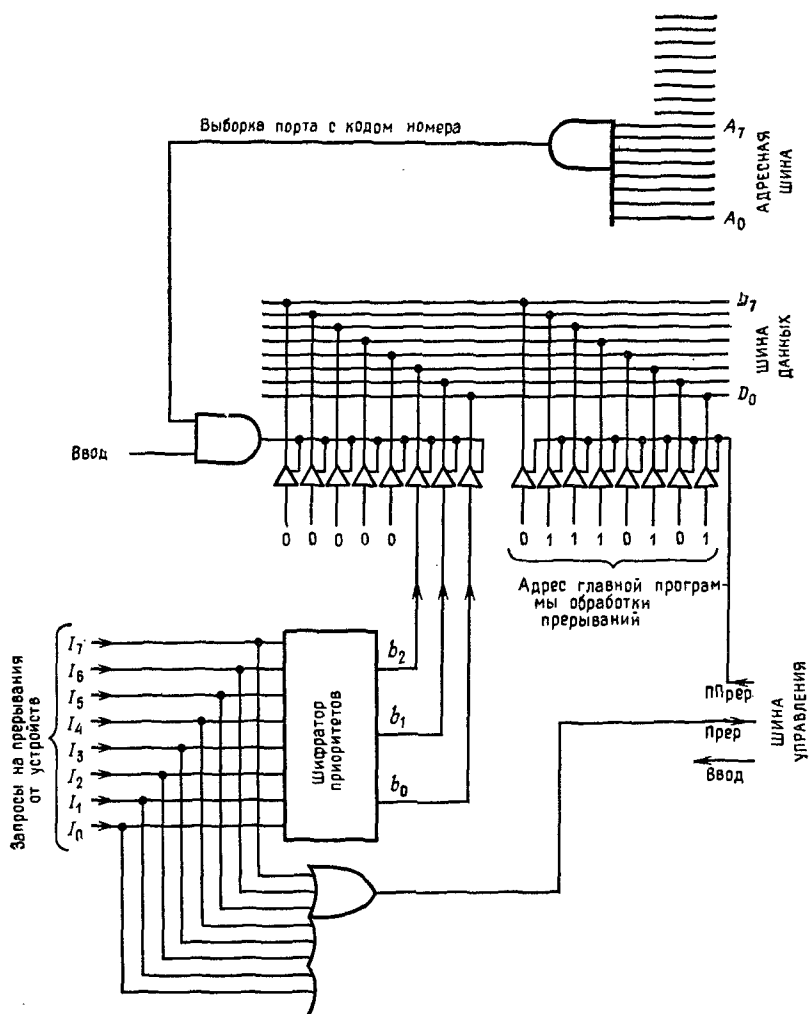


Рис. 27.7. Векторная система прерываний для иллюстративного микропроцессора с выбором прерывающего устройства при помощи шифратора приоритетов.

ввода и тем самым становится доступным для микропроцессора.

Запросы на прерывания от устройств подаются также на входы вентиля ИЛИ, выход которого подключен к линии «Прер» микропроцессора. Таким образом, микропроцессор получит сигнал прерывания, если есть запрос хотя бы от одного устройства. На сигнал «Прер» микропроцессор в соответствующем

щий момент ответит сигналом «ППер». Этот сигнал служит для подачи фиксированного начального адреса главной программы обработки прерываний на шину данных для последующей передачи его в микропроцессор.

После этого начинается выполнение главной программы обработки прерываний. Прежде всего эта программа вводит код номера старшего устройства из числа запросивших прерывание. Далее оно может поступить одним из двух способов. Она может воспользоваться полученным номером для перехода на программу, обслуживающую только данное устройство. Или же она может перейти на программу, общую для многих устройств, которая воспользуется полученным номером для определения кода порта, соответствующего обслуживаемому устройству. Вторая возможность особенно удобна, когда устройства допускают однотипное обслуживание.

Как упоминалось выше, такая система прерываний может работать с любым микропроцессором. Это действительно так, поскольку для передачи в микропроцессор информации о прерывающем устройстве не требуется ничего, кроме порта ввода.

Векторная система прерываний с начальной командой, поступающей от устройства

В некоторых микропроцессорах применяется система прерываний с посылкой в микропроцессор команды от устройства. Такие микропроцессоры в ответ на поступившее прерывание выбирают очередную выполняемую команду не из памяти, а из схем интерфейса с соответствующим устройством. В отличие от других систем прерываний в данной системе не нужно автоматически сохранять программный счетчик.

Иногда единственной поступающей от устройства команды оказывается достаточно, чтобы обработать прерывание. Например, при программной реализации счетчика событий нужно лишь подсчитывать число прерываний от некоторого конкретного источника. Для этого достаточно одной команды, увеличивающей на 1 содержимое регистра или ячейки памяти. В таких случаях, выполнив единственную команду, обрабатывающую прерывание, процессор возобновляет выполнение основной программы.

Однако в большинстве случаев одной команды недостаточно, чтобы обработать прерывание. В таком случае устройство может послать в микропроцессор команду перехода на подпрограмму, которая и выполнит обработку прерывания.

Существует множество других систем прерывания программ. Однако, как правило, они представляют собой разновидности тех схем, которые мы рассмотрели.

Программные аспекты прерывания программы

Интересно сравнить механизмы прерывания программы и вызова подпрограммы. В обоих случаях происходит приостановка основной программы на время выполнения вспомогательной. Следовательно, в обоих случаях должна быть организована связь основной программы со вспомогательной, обеспечивающая возобновление работы первой после завершения второй. При организации этой связи нужно предусмотреть сохранение программного счетчика в таком месте, откуда он может быть восстановлен, например в стеке или в памяти вспомогательной программы.

Тем не менее факт, что вход в программу обработки прерывания вызывается внешним событием, а вход в подпрограмму — командой перехода в «главной» программе, приводит к важным отличиям между программой обработки прерывания и подпрограммой. Точки входа в подпрограмму находятся полностью под контролем программиста. В частности, ему точно известны регистры и флаги, используемые в программе в момент вызова подпрограммы, и, следовательно, он может сохранить содержимое тех регистров и флагов, которые используются также и в подпрограмме.

Прерывание программы, с другой стороны, может произойти в любой точке, и, следовательно, сохраняться должны все регистры и флаги, используемые в программе обработки прерывания. В свете этих отличий в некоторых микропроцессорах все регистры и флаги автоматически сохраняются при прерываниях в стеке или в особой рабочей области памяти и не сохраняются автоматически при вызове подпрограммы. Для таких процессоров нужны две команды возврата: одна, восстанавливающая все регистры и флаги, включая программный счетчик, — для прерываний и другая, восстанавливающая только программный счетчик, — для подпрограммы.

Автоматическое сохранение всех регистров имеет как достоинства, так и недостатки. Оно уменьшает число команд, затрачиваемых на сохранение регистров, в программе, обслуживающей прерывание. Однако это не всегда приводит к экономии времени и даже может приводить к необоснованным его тратам, когда регистры и флаги сохранять не нужно.

В иллюстративном микропроцессоре такого различия между механизмами прерывания программы и обращения к подпрограмме не проводится. В обоих случаях в стеке автоматически сохраняется только программный счетчик. Содержимое первых трех общих регистров и флаг переноса *C* можно сохранить и в дальнейшем восстановить командами *PUSH* и *POP*. Вообще говоря, этих трех регистров должно быть достаточно для рабо-

ты программы прерывания. Прежде чем пользоваться каким-либо из этих регистров или триггером переноса, программа обработки прерывания должна их сохранить в стеке, а после выполнения своей задачи — восстановить из стека.

В качестве примера рассмотрим программу обработки прерывания для иллюстративного микропроцессора, решающего следующую несложную задачу. Через устройство ввода-вывода в микропроцессор поступают 8-битовые целые положительные числа. Микропроцессор сравнивает каждое из них с константой, скажем $5B_{16}$, и выводит только те из них, которые оказываются меньше этой константы. Предполагается, что процессор выполняет также и другие вычисления, так что эта задача должна использовать прерывания. Будем также предполагать, что других прерываний не будет.

Программа обработки прерывания для такого режима работы приведена в табл. 27.2. Когда устройство ввода загружа-

Таблица 27.2. Пример программы обработки прерывания

Ячейка памяти	Команда на машинном языке	Команда в символической форме	Комментарий
0080	77	PUSH	Сохранение Асс, Н, L и С в стеке
0081	FD	INP	Ввод числа
0082	12	12	
0083	61	LRI 1	Загрузка константы 5B в Н
0084	5B	5B	
0085	A1	SUB 1	Вычитание 5B из числа
0086	7C	JCZ	Проверка знака разности; обход вывода, если разность положительна
0087	00	00	
0083	8C	8C	
0089	81	ADD 1	+5B для восстановления числа
008A	FE	OUT	Вывод числа
008B	12	12	
008C	73	POP	Восстановление Асс, Н, L, С
008D	FB	EIT	Снятие блокировки прерываний
008E	F8	RET	Возврат в прерывную программу

ет очередное число в порт ввода, происходит прерывание и начинается выполнение программы, обрабатывающей прерывания. Первая команда PUSH сохраняет содержимое трех регистров и триггера переноса. Следующая команда вводит число (предполагается, что порт ввода имеет номер 12_{16}) и загружает его в аккумулятор. Затем производится вычитание константы 5B с предварительной загрузкой ее в регистр,

Если введенное число меньше константы, то разность будет отрицательной и на триггере переноса установится 1. Если это не так, то команда «переход при нулевом переносе» (JCZ) обходит две следующие за ней команды. Эти две команды сначала снова прибавляют к аккумулятору константу 5В, чтобы восстановить значение введенного числа, а затем выводят его (порт вывода имеет тот же номер 12₁₆). После этого командой ROP восстанавливаются первые три регистра и триггер переноса. Перед возвратом на прерванную программу устанавливается в 1 флаг прерываний (который был сброшен автоматически при входе в программу прерывания), в результате чего блокировка прерываний снимается. И наконец, последняя команда возвращает управление на прерванную программу.

27.4. ИНТЕРФЕЙС С ГЛАВНОЙ ПАМЯТЬЮ

Главная память — это по существу основная часть микрокомпьютера, поскольку в ней хранятся программы и данные. В некоторых микрокомпьютерах для хранения команд и данных используются отдельные запоминающие устройства. Они могут отличаться длиной слова, способами адресации, методами доступа и скоростью работы. Однако в большинстве микрокомпьютеров команды и данные размещаются в одной и той же памяти. Рассуждения этого раздела можно относить как к раздельным ЗУ, так и к общей памяти.

Вообще говоря, главная память микрокомпьютера представляет собой память с произвольным доступом, состоящую из слов с некоторым числом битов в каждом. Каждое слово имеет свой адрес, по которому процессор может обратиться именно к этому слову. В большинстве микрокомпьютеров главная память состоит из модулей; каждый из них представляет собой одну интегральную схему. Интегральные схемы соединяются друг с другом так, чтобы слова имели нужный размер и разные адреса.

В принципе в рамках одной главной памяти могут объединяться модули как постоянной памяти (ПЗУ), так и памяти со считыванием и записью (ОЗУ). Некоторые ячейки могут быть отведены для постоянных программ и констант и реализованы в виде ПЗУ, тогда как другие, отведенные для изменяющихся данных, могут быть реализованы в виде ОЗУ. Может оказаться, что некоторые или все модули имеют разрядность меньше той, которая нужна для данного микропроцессора. В этом случае модули объединяются так, чтобы их суммарная разрядность соответствовала нужной и чтобы они работали в параллель,

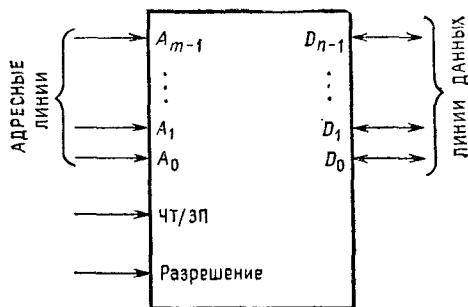


Рис. 27.8. Модель модуля статической памяти.

Важное значение имеют временные задержки при обращении процессора к главной памяти. Благодаря относительно более высокой степени интеграции модулей ЗУ они обычно работают медленнее электронных компонент, применяемых в портах ввода-вывода. Поэтому, для того чтобы модули ЗУ можно было применять совместно с некоторым конкретным процессором, они должны обладать определенными временными характеристиками.

Модель модуля памяти

Интерфейс модуля памяти с микропроцессором характеризуется главным образом временными параметрами и адресацией. Чтобы разобраться в этом интерфейсе, мы построим модель большинства ОЗУ статического типа.

На рис. 27.8 представлены внешние сигнальные линии нашей модели. В ней m входных линий задают адрес слова, а n двунаправленных линий для данных соответствуют числу битов в каждом слове. Таким образом, модуль содержит 2^m слов по n битов в каждом. Для управления операциями записи и чтения служат две входные линии ЧТ/ЗП и «разрешение».

Модуль находится в режиме чтения, если сигнал на ЧТ/ЗП равен логической 1; в противном случае он находится в режиме записи. Чтобы была выполнена какая-то операция, на линию «разрешение» должна быть подана логическая 1. Пока на линии разрешения 0, линии данных находятся в третьем (свободном) состоянии. Для модулей ПЗУ линия ЧТ/ЗП отсутствует и считается, что модуль постоянно находится в режиме чтения. Реальные модули в некоторых аспектах могут отличаться от описанной модели, а именно:

- 1) Могут существовать дополнительные линии разрешения, иногда называемые **линиями выборки модуля**, которые внутри

модуля объединяются с основными линиями разрешения с помощью вентилей И.

2) Двухнаправленные линии данных могут быть заменены раздельными входными и выходными линиями.

3) Линия «разрешение» может быть заменена двумя, из которых одна управляет чтением, а другая — записью.

4) Формирователи, работающие на линии данных, могут оказаться не тристабильными, а с открытым коллектором или эмиттером.

Тем не менее эти вариации не должны оказать серьезного влияния на рассматриваемый ниже интерфейс модулей ЗУ с микропроцессором.

Пространство памяти микропроцессора

Процесс компоновки памяти микропроцессора из отдельных модулей можно представлять себе как процесс заполнения некоторого «пространства». Пространство памяти микропроцессора можно считать прямоугольной областью, разбитой на множество строк, соответствующих адресам. По горизонтали в каждой строке располагаются биты, составляющие слово памяти. Например, в нашем иллюстративном микропроцессоре пространство памяти состоит из 2^{16} строк по 8 бит в каждой. Модули запоминающих устройств различного типа и размеров размещаются в пространстве памяти в соответствии с требованиями конкретного приложения. На рис. 27.9 приведено одно из возможных таких размещений.

После того как принято решение о размещении модулей в пространстве памяти, нужно определить необходимые связи. Линии данных всех модулей подсоединяются к соответствующим линиям шины данных микропроцессора. С помощью адресных линий микропроцессора, во-первых, выбирается модуль или группа параллельно работающих модулей и, во-вторых, ячейка внутри выбранного модуля или модулей. Поскольку слова в одном модуле обычно соответствуют последовательным словам пространства памяти, адресные линии модуля (т. е. его внутренние адреса) подсоединяются к младшим адресным линиям микропроцессора. Остальные адресные линии микропроцессора используются для выборки нужного модуля. Выборка модуля в нашем случае осуществляется при помощи управляющей линии «разрешение».

Линия «разрешение» вместе с линией ЧТ/ЗП используется также для определения момента выполнения операций и ее вида. К линиям «разрешение» и ЧТ/ЗП подключаются управляющие линии микропроцессора.

При операции чтения на линию ЧТ/ЗП всех модулей подается логическая 1, и, кроме того, в соответствующий момент

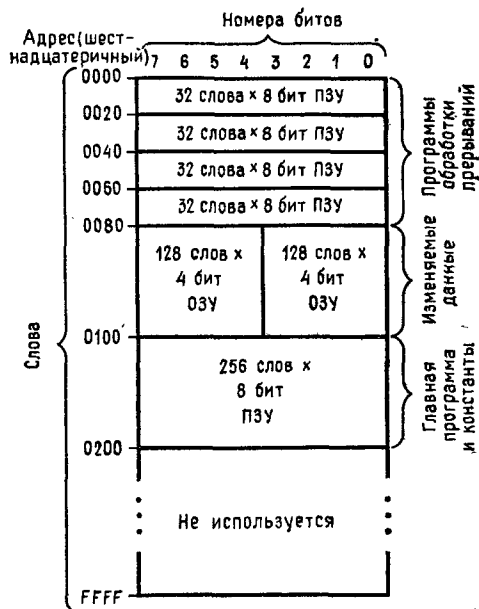


Рис. 27.9. Пример размещения модулей в пространстве памяти иллюстративного процессора.

логическая 1 подается на линию «разрешение» выбранного модуля. При этом на шине данных появляется содержимое адресуемой ячейки и оно передается в микропроцессор. При операции записи на линии ЧТ/ЗП всех модулей подается логический 0, и, кроме того, в соответствующий момент подается логическая 1 на линию «разрешение» выбранного модуля. Это приводит к записи в адресуемую ячейку выбранного модуля или модулей той информации, которую подал микропроцессор на шину данных.

Пример компоновки памяти

Схема подключения модулей памяти к микропроцессору показана на рис. 27.10. Она соответствует размещению, приведенному на рис. 27.9. Всего на схеме семь модулей, отличающихся размерами и типом. Во всех модулях адресные линии соединены с младшими линиями адресной шины, а линии данных — с соответствующими линиями шины данных. Для формирования сигналов на линии «разрешение» берутся комбинации сигналов от нужных линий адресной шины, а также управляющий сигнал «чтение» для модулей ПЗУ или логическая сумма

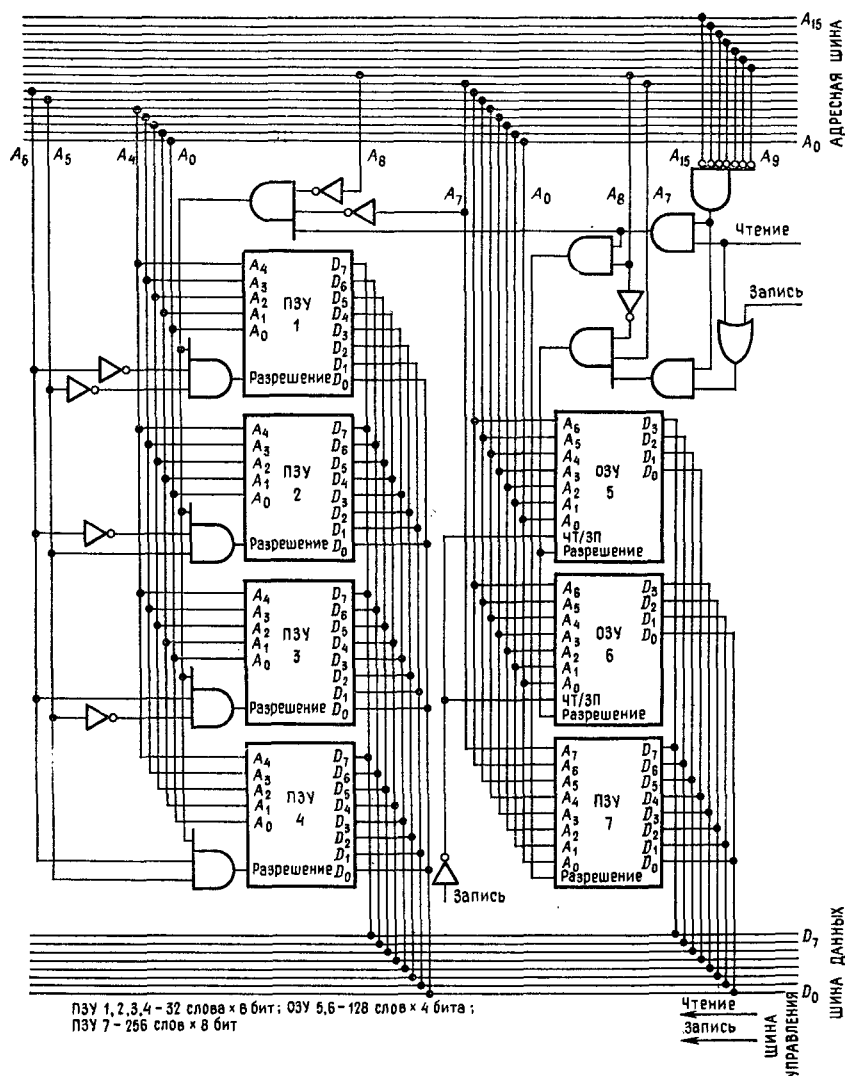


Рис. 27.10. Схема подключения модулей памяти к иллюстративному микропроцессору в соответствии с размещением, показанным на рис. 27.9.

сигналов «чтение» и «запись» для модулей ОЗУ. Таким образом, модуль получает сигнал разрешения только в том случае, когда на адресную шину подана соответствующая комбинация и когда должна произойти операция чтения или записи.

Точнее, условия, при которых дается сигнал разрешения на каждый модуль, можно описать с помощью логических выражений. Обозначим через E_i сигнал разрешения для модуля i , а через A_j сигнал на линии j адресной шины. Заметим, что во все сигналы разрешения входит логическое произведение инверсий старших семи линий адресной шины. В сигналы от E_1 до E_4 , кроме того, входят сигналы «чтение», \bar{A}_7 , \bar{A}_8 и некоторая комбинация из A_5 и A_6 . Эти четыре сигнала задаются следующими выражениями:

$$E_1 = \text{„чтение“ } \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9\bar{A}_8\bar{A}_7\bar{A}_6\bar{A}_5,$$

$$E_2 = \text{„чтение“ } \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9\bar{A}_8\bar{A}_7\bar{A}_6A_5,$$

$$E_3 = \text{„чтение“ } \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9\bar{A}_8\bar{A}_7A_6\bar{A}_5,$$

$$E_4 = \text{„чтение“ } \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9\bar{A}_8\bar{A}_7A_6A_5.$$

В сигналы E_5 и E_6 входит «чтение» + «запись» \bar{A}_7 и \bar{A}_8 . Сигналы E_5 и E_6 равны между собой и определяются выражением

$$E_5, E_6 = (\text{„чтение“} + \text{„запись“}) \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9\bar{A}_8\bar{A}_7.$$

Наконец, E_7 зависит от «чтение» и A_8 и задается выражением

$$E_7 = \text{„чтение“ } \bar{A}_{15}\bar{A}_{14}\bar{A}_{13}\bar{A}_{12}\bar{A}_{11}\bar{A}_{10}\bar{A}_9A_8.$$

Следует заметить, что все сигналы E_i , кроме сигналов E_5 и E_6 , относящихся к параллельно работающим модулям, взаимоисключающие. Схему на рис. 27.10 можно упростить, если известно, что память не будет расширяться. В этом случае общий множитель $\bar{A}_{15} \dots \bar{A}_9$ можно убрать из всех E_i . В результате окажется, что к одной и той же ячейке можно будет обращаться еще по нескольким адресам, кроме ее «основного» адреса. Как правило, это не вызывает никаких трудностей и может даже использоваться при программировании.

Временные параметры памяти

Время, затрачиваемое на передачу данных в память и из памяти, как правило, играет важную роль, поскольку память работает медленнее электронных компонентов малой степени интеграции, используемых в портах ввода-вывода. Для правильного выполнения операций чтения и записи нужно учитывать возникающие задержки.

При операции чтения, вообще говоря, сначала на модуль памяти должен быть подан адрес, затем подана логическая 1 «разрешения» и, наконец, через определенное время прочитанные данные могут быть взяты с линий шины данных. Главной

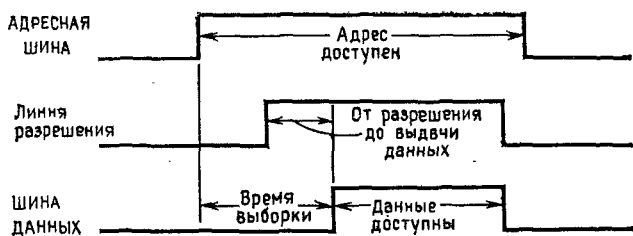


Рис. 27.11. Временные соотношения при выполнении операции чтения в модуле памяти.

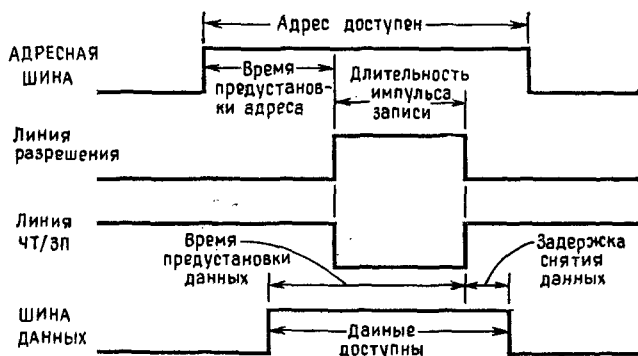


Рис. 27.12. Временные соотношения при выполнении операции записи в модуле памяти.

характеристикой здесь является время задержки между моментом подачи адреса и моментом получения данных. Это время называется **временем выборки**, и микропроцессор должен его учитывать в своей работе, иначе опрос шины данных может быть сделан преждевременно, т. е. до поступления на нее правильных данных. Также существенным является время между моментом установления логической 1 на линии «разрешение» и поступлением информации на шину данных. Это время обычно меньше времени выборки и также должно учитываться микропроцессором. Временные соотношения при чтении показаны на рис. 27.11.

Запись в память сложнее чтения и поэтому требует рассмотрения большего числа временных соотношений. При записи на входы модуля памяти нужно подать адрес и данные и, поддерживая состояние этих входов постоянным, подать кратковременные сигналы логической 1 и логического 0 на линии «разрешение» и ЧТ/ЗП соответственно. При этом существенное значение имеют несколько временных интервалов, показанных на рис. 27.12,

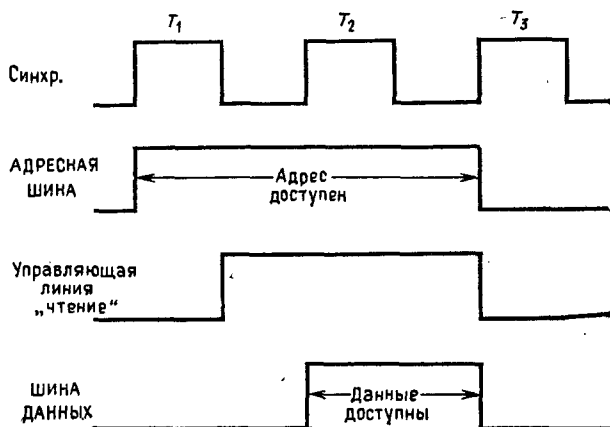


Рис. 27.13. Временная диаграмма работы иллюстративного микропроцессора при чтении из памяти.

После подачи адреса должно пройти время, необходимое для срабатывания внутренних декодирующих схем. Если при наличии логической 1 на линии разрешения будет подан 0 на линию ЧТ/ЗП до того, как завершится декодирование адреса, то может произойти запись в ячейку с неверным адресом. Время между подачей адреса и началом сигнала «запись» часто называют **временем предустановки адреса**.

Данные, поданные на вход модуля, записываются в конкретную ячейку в интервале, когда сигнал ЧТ/ЗП равен 0, а сигнал разрешения равен 1. Этот интервал, характеризующий **длительность импульса записи**, должен быть достаточным для регистрации данных на запоминающих элементах. Считается, что в адресуемую ячейку записываются данные, которые находятся на линиях шины данных в момент прохождения заднего фронта «импульса записи», но эти данные должны быть установлены за некоторое время до заднего фронта и должны сохраняться на шине некоторое время после него. Первое время называется **временем предустановки данных**, второе — **временем задержки снятия данных**.

При рассмотрении временных соотношений должны учитываться дополнительные задержки во вспомогательных вентилях, входящих в интерфейс. Однако этими задержками часто можно пренебречь из-за относительно более высокого быстродействия элементов с малой степенью интеграции.

Рассмотрим теперь, каким образом описанные временные соотношения должны соблюдаться со стороны микропроцессора. Сначала рассмотрим временную диаграмму, показанную

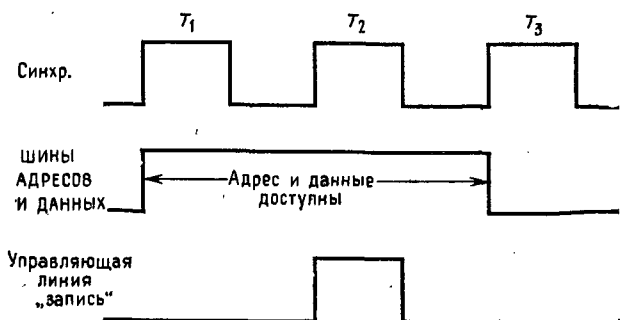


Рис. 27.14. Временная диаграмма работы иллюстративного микропроцессора при записи в память.

на рис. 27.13, для операции чтения. На ее выполнение затрачивается машинный цикл из трех синхронимпульсов. Микропроцессор работает в предположении, что данные на шине данных доступны между передними фронтами импульсов T_2 и T_3 . Время от переднего фронта T_1 , когда на адресную шину подается адрес, до переднего фронта T_2 и составляет время выборки.

Поданный адрес определяет модуль памяти (или группу параллельно работающих модулей) и ячейку внутри модуля (или модулей). По истечении некоторого времени по заднему фронту T_1 возбуждается управляющая линия «чтение» микропроцессора, что приводит к подаче сигнала «разрешение» на выбранный модуль. В промежуток между импульсами T_1 и T_2 входит время от разрешения до выдачи данных плюс задержка на формирование сигнала «разрешение».

При чтении линии ЧТ/ЗП во всех модулях содержат логическую 1, поскольку они управляются инверсией сигнала «запись», поступающего от микропроцессора. Следовательно, содержимое адресуемой ячейки выбранного модуля поступит на шину данных по прошествии некоторого времени после прихода сигнала «разрешение». Стробирование передачи данных на соответствующий регистр микропроцессора выполняется по заднему фронту T_2 .

Временная диаграмма для операции записи в иллюстративном микропроцессоре показана на рис. 27.14. Стробирование передачи данных в выбранную ячейку выполняется в течение синхроимпульса T_2 . Время от переднего фронта T_1 , когда на соответствующие шины подаются адрес и данные, до переднего фронта T_2 составляет время предустановки адреса, а до заднего фронта T_2 — время предустановки данных.

Импульс на управляющей линии микропроцессора «запись» по времени совпадает с T_2 . Этот импульс управляет работой

схемы формирования сигналов «разрешение» и обеспечивает подачу разрешения на модуль, которому принадлежит ячейка с заданным адресом. Импульс «запись» вызывает также подачу логического 0 на линию ЧТ/ЗП в модули ОЗУ, задавая тем самым режим записи. Таким образом, длительность импульса записи отличается от длительности синхроимпульсов только на время задержки, которую вносят вентили схем разрешения. Адрес и данные сохраняются на шинах до переднего фронта T_3 , в результате чего интервал между импульсами T_2 и T_3 учитывает время задержки снятия данных, необходимое для работы ЗУ.

Все упоминавшиеся выше временные требования со стороны модулей памяти должны быть удовлетворены. Если хотя бы одно из них оказывается неудовлетворенным, должна быть соответствующим образом снижена частота тактовых импульсов.

Здесь полезно заметить, что временные диаграммы различных микропроцессоров отличаются друг от друга весьма существенно. Некоторые микропроцессоры спроектированы специально для работы с запоминающими устройствами определенного типа, имеющими свой особый «профиль» временных соотношений. Использование в таком микропроцессоре ЗУ другого типа может привести к очень большому снижению частоты синхронизации ради удовлетворения всего одного какого-нибудь требования. Некоторые микропроцессоры, например, спроектированы в расчете на память практически с нулевым временем задержки снятия данных. Применение с таким микропроцессором памяти с достаточно большим значением этого параметра может привести к чрезмерному снижению частоты. Иногда, для того чтобы справиться с описанными трудностями, используют источники задержанных синхроимпульсов или в микропроцессоре, или в памяти.

Системы динамической памяти

Основное отличие динамической памяти от статической связано с необходимостью периодически регенерировать информацию. Вообще говоря, регенерация происходит, когда выполняется операция чтения или записи. Рассмотрим частный вид модуля памяти, в котором регенерация осуществляется при чтении. Модель такого модуля по сравнению с моделью статического модуля содержит одну дополнительную управляющую линию, показанную на рис. 27.15. Совместное использование двух управляющих линий «разрешение» и «выборка» позволяет выполнить функции, аналогичные функциям линии «разрешение» в статическом модуле. Линия «разрешение» выполняет внутренние управляющие функции в модуле, тогда как линия

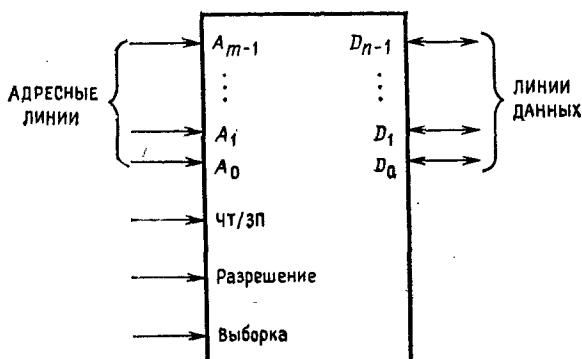


Рис. 27.15. Модель модуля динамической памяти.

«выборка» управляет линиями данных, переводя их в третье состояние при логическом 0. При чтении и записи на обе линии сигнал поступает одновременно. Однако при регенерации возбуждается только линия «разрешение».

Каждая операция регенерации подновляет содержимое нескольких ячеек в модуле. Внутри модуля слова образуют матрицу из строк и столбцов. Адрес слова состоит из двух частей: адреса строки и адреса столбца. Биты одного слова можно считать расположенными как бы в третьем измерении. Все слова в заданной строке регенерируются одновременно.

Операция регенерации сводится к заданию адреса строки и последующей подаче сигнала на линию «разрешение» при значении 1 на линии ЧТ/ЗП. Линии адреса столбца не оказывают влияния на регенерацию и могут содержать любые значения. Каждая строка в модуле динамической памяти должна регенерироваться в пределах ограниченного промежутка времени после предыдущей регенерации. Этот промежуток (период регенерации) обычно порядка нескольких миллисекунд. За период регенерации должно быть выполнено столько операций регенерации, каково число строк в модуле памяти.

Поскольку в момент регенерации обращаться к памяти со стороны микропроцессора нельзя, должны существовать средства приостановки работы микропроцессора на время регенерации. В разных процессорах эта задача решается разными методами. В некоторых микропроцессорах используется специальный управляющий сигнал, приостанавливающий работу микропроцессора и дающий возможность выполнить регенерацию. Другую возможность дают средства прямого доступа к памяти, которые будут рассмотрены в следующем разделе. Еще одну возможность провести регенерацию можно получить, крат-

ковременно воздействуя на подаваемые в микропроцессор тактирующие импульсы.

Во всех вариантах работа микропроцессора приостанавливается и процедура регенерации осуществляется внешними управляющими схемами. Эти схемы определяют как момент выполнения, так и адрес строки для каждой операции регенерации.

Пример схемы регенерации

Одна из возможных схем регенерации, которую можно было бы применить в иллюстративном микропроцессоре для динамической памяти, приведена на рис. 27.16. Для приостановки микропроцессора на время регенерации блокируется линия тактирования. Это происходит перед началом каждого машинного цикла, когда адресная шина предполагается установленной микропроцессором в третье состояние.

На схеме рис. 27.16 начало каждого машинного цикла определяется с помощью выходной управляющей линии микропроцессора «синхр». На эту линию подается импульс, совпадающий по времени с третьим тактовым импульсом каждого машинного цикла. Время операции регенерации определяется при помощи k -разрядного счетчика, увеличивающегося по каждому импульсу в линии «синхр». К выходам триггеров этого счетчика подключен k -входовый вентиль И, на выходе которого появится логическая 1, если единицы будут на всех k входах. Это условие определяет момент проведения регенерации. Очередной тактовый импульс не пропускается на микропроцессор и используется для регенерации.

Для выполнения регенерации в модуле памяти на линию адреса строки подается содержимое 4-битового счетчика через тристабильные формирователи. Затем возбуждается линия «разрешение» импульсом от тактового генератора, проходящим через два вентиля. Фактически этим же импульсом регенерации увеличивается счетчик адреса строки для следующей операции регенерации. Кроме того, этим же импульсом продвигается и k -разрядный счетчик, который принимает нулевое значение и прекращает операцию регенерации.

Следующий синхрои́мпульс от тактового генератора будет пропущен на микропроцессор в качестве первого импульса нового машинного цикла. После этого k -разрядный счетчик снова будет подсчитывать импульсы «синхр», отмеряя время до очередной операции регенерации. Нужно отметить, что методы регенерации динамической памяти не только меняются от микропроцессора к микропроцессору, но меняются даже для разных приложений микропроцессоров одного и того же типа. Поэтому описанную схему следует воспринимать лишь как иллюстрацию

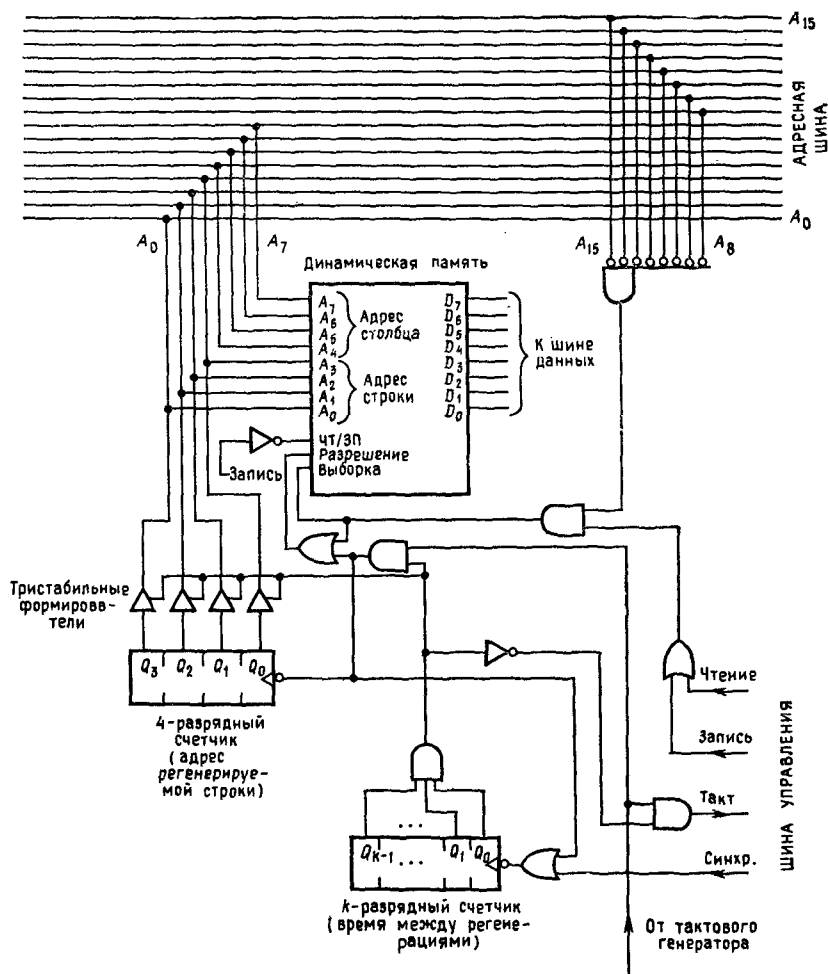


Рис. 27.16 Схема регенерации динамической памяти для иллюстративного микропроцессора.

в самом общем смысле. Проектирование реальной схемы регенерации требует тщательного учета специфики конкретного микропроцессора, модулей памяти и характера приложений.

27.5. ПРЯМОЙ ДОСТУП К ПАМЯТИ

До сих пор в этой главе мы рассматривали передачи данных между микропроцессором и внешними устройствами или между микропроцессором и главной памятью. Однако интерес

представляют также прямые передачи между внешними устройствами и главной памятью. Такие передачи можно было бы использовать, например, для начальной загрузки программы в память из устройства ввода. В процессе работы программы данные также могли бы вводиться из устройства в память для последующей обработки или результаты, получаемые микропроцессором, могли бы накапливаться в памяти для последующей выдачи на внешнее устройство.

Когда в микрокомпьютере существуют средства передачи данных между памятью и внешним устройством без непосредственного вмешательства программы в процесс передачи, они называются средствами **прямого доступа к памяти (ПДП)**. На самом деле, добавив достаточное количество вспомогательных схем, в любом микрокомпьютере можно обеспечить прямой доступ к памяти. Однако, для того чтобы свести число информационных каналов к минимуму, в большинстве микропроцессорных систем предусматриваются специальные средства, позволяющие использовать обычные шины для прямого доступа к памяти. Это достигается тем, что на время ПДП-передачи управление шинами забирается из-под контроля микропроцессора и передается внешнему устройству, которое может воспользоваться шинами для передачи данных между устройством и памятью.

Выполняемая программа, поскольку она тоже нуждается в обменах с памятью, как правило, приостанавливается на время «захвата» шин внешним устройством. Хотя существуют некоторые вариации между различными микропроцессорами, освобождение шин со стороны процессора, как правило, происходит при поступлении запроса на ПДП со специальной управляющей линии. В нашем иллюстративном микропроцессоре предусмотрены две управляющие линии ПДП и ППДП; по одной из них подается запрос на прямой доступ к памяти, а по другой — ответ на запрос (подтверждение).

Если по линии ПДП извне в микропроцессор приходит логическая 1, то после завершения очередного машинного цикла процессор приостанавливает выполнение программы и освобождает адресную шину и шину данных, переводя их линии в третье (свободное) состояние. После этого микропроцессор выдает подтверждение в виде логической 1 на линию ППДП. Все это время процессор сохраняет контроль и над другими выходными линиями шины управления. В противном случае неопределенное состояние управляющих линий могло бы вызывать нежелательные события, например запись в случайную ячейку памяти.

В частности, во время подачи 1 на линию ППДП на управляющие линии «чтение» и «запись» микропроцессор подает

логической 0. Обычно эти линии подсоединяются прямо к памяти. При наличии прямого доступа нужны соответствующие изменения в схемах, которые дали бы возможность управлять чтением и записью со стороны внешнего устройства. Достаточно, например, объединить с помощью вентилях ИЛИ линии «чтение» и «запись» от микропроцессора с аналогичными линиями от внешнего устройства, а выходы вентилях соединить с соответствующими линиями памяти.

Устройство, желающее получить прямой доступ к памяти, сначала устанавливает логическую 1 на линии ПДП. Затем оно ждет, пока появится логическая 1 на линии ППДП. После этого устройство использует адресную шину, шину данных и свои линии чтения и записи для обмена данными с памятью аналогично тому, как это делает микропроцессор. Осуществив все необходимые передачи, устройство устанавливает на линии ПДП значение 0, в результате чего микропроцессор снова получает управление шинами.

Передачу данных в режиме прямого доступа к памяти можно считать одной из форм ввода-вывода, отличающегося от программируемого ввода-вывода тем, что управляет передачей не программа, а внешние схемы. При прямом доступе к памяти экономится время. Кроме того, прямой доступ к памяти в некоторых случаях может оказаться единственным способом подключения таких быстрых внешних устройств, как магнитные диски или ленты, когда программируемый ввод-вывод может просто не успевать за потоком данных.

Передача блоков данных с использованием ПДП

Быстрые устройства, подключаемые с использованием прямого доступа к памяти, обычно передают данные блоками слов. Программа, выполняемая микропроцессором, как правило, инициирует передачу блока и задает его размер. После этого отдельные слова передаются под управлением схем независимо от программы. Предположим, в программе потребовался ввод некоторого количества слов с магнитной ленты в память. В этом случае программа могла бы содержать команды, с помощью которых на внешние схемы, управляющие прямым доступом, было бы передано число вводимых слов и начальный адрес области памяти, в которую их нужно ввести. Затем программа могла бы установить флаг, запускающий работу схемы ПДП. С этого момента программа могла бы выполнять другую работу, предоставив управление передачей внешним схемам.

На рис. 27.17 приведена схема для иллюстративного микропроцессора, которая управляет блочной передачей данных из устройства ввода в режиме прямого доступа. Схема содержит

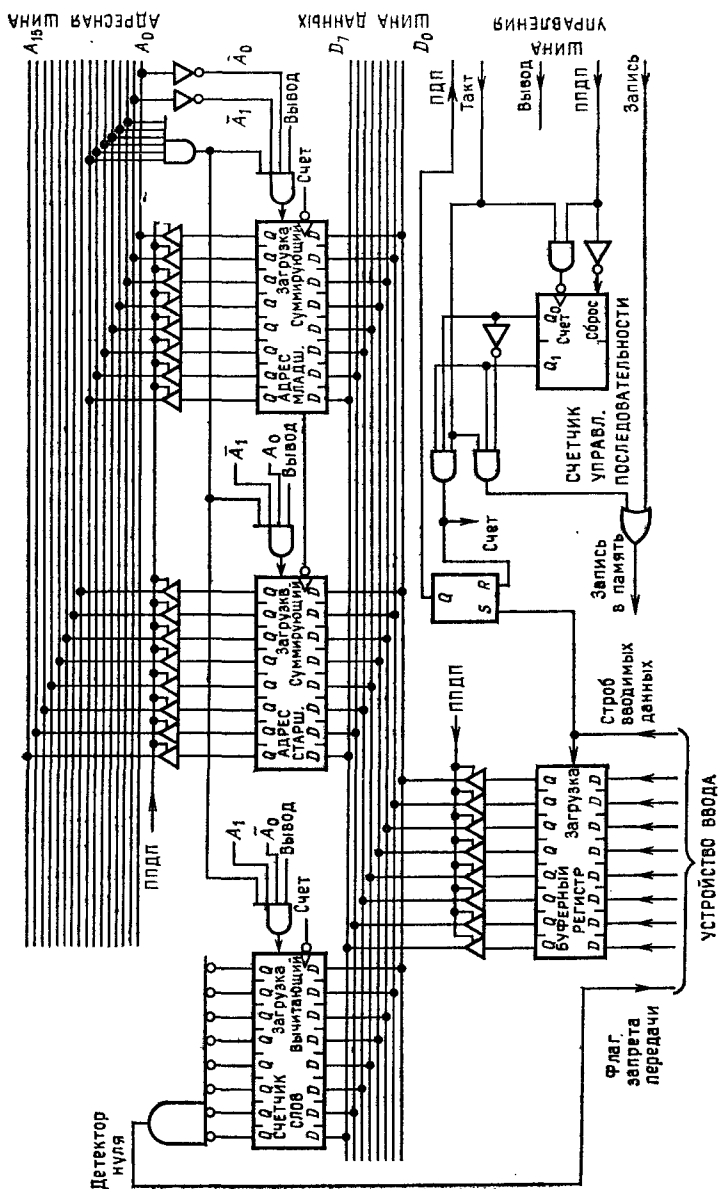


Рис. 27.17. Схема управления передачей данных из внешнего устройства в режиме прямого доступа к памяти.

8-битовый буферный регистр для приема данных от устройства и выдачи их на шину данных. Текущий адрес памяти, по которому происходит запись данных, хранится на двух 8-битовых суммирующих счетчиках (старшая и младшая половина адреса). Счетчики соединены последовательно и работают как один 16-битовый счетчик. Для подсчета числа переданных слов используется восьмибитовый вычитающий счетчик. Оба адресных счетчика и счетчик слов подключены к микропроцессору как регистры портов вывода.

Программа инициирует передачу блока с посылки команды вывода на адресные счетчики младшей и старшей половины 16-битового адреса, а на счетчик слов 8-битового числа. К выходам счетчика слов подключен вентиль И с инверсными входами, который служит детектором нуля на счетчике. До тех пор пока на счетчике не ноль, на выходе детектора — логический 0, и это разрешает устройству продолжать ввод данных. Устройство в ответ посылает вводимые данные слово за словом, получая их с носителя, например с магнитной ленты.

Из устройства поступает сигнал, стробирующий прием данных на буферный регистр. Тот же стробирующий импульс устанавливает 1 на триггере, выход которого подается на линию ПДП в микропроцессор. Получив сигнал ПДП, микропроцессор приостанавливает выполнение программы в конце очередного машинного цикла и освобождает шины. На выходную управляющую линию ППДП при этом подается 1, что вызывает запуск последовательности из 4 импульсов, в которой содержимое буферного регистра передается в память. Управляет этой последовательностью двухбитовый счетчик, на счетный вход которого подаются тактовые импульсы, при условии ППДП = 1.

Значение 1 на линии ППДП обеспечивает подачу содержимого двух адресных счетчиков на адресную шину, а буферного регистра — на шину данных. Каждый тактовый импульс продвигает двухбитовый счетчик. По первым двум импульсам больше никаких действий не выполняется, но по третьему импульсу в память посылается сигнал записи. Он формируется на двух вентилях. На первый вентиль, вентиль И, подается тактирующий импульс и комбинация 10 двухбитового счетчика. На второй вентиль, вентиль ИЛИ, подается выход вентиля И и линия «запись» микропроцессора. Импульс записи обеспечивает запись очередного введенного слова в память по адресу на адресной шине.

Следующий за импульсом записи тактовый импульс переводит двухбитовый счетчик в состояние 00 и возбуждает импульс на линии «счет». Этот импульс увеличит адрес на адресных счетчиках и уменьшит счетчик слов. Он же сбросит триггер, подсоединенный к линии ПДП. Таким образом, теперь счетчи-

ки готовы к передаче следующего слова, и микропроцессор получает разрешение возобновить выполнение программы. Когда счетчик слов дойдет до 0, на выходе детектора нуля появится логическая 1, и это будет означать, что передача блока данных завершена. Устройство прекратит передачу данных до тех пор, пока микропроцессор не инициирует передачу нового блока.

Подобного типа передачи часто называют передачами с **захватом цикла**, поскольку они задерживают выполнение программы примерно на один машинный цикл. Другой тип передачи вызывает полную остановку программы на все время передачи блока.

27.6. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О ШИНАХ

Рассмотренная выше общая структура шин в основных чертах соответствует большинству микропроцессоров и большинству приложений микропроцессоров. Тем не менее имеет смысл остановиться еще на нескольких понятиях, которые могут встретиться при рассмотрении некоторых микропроцессоров или некоторых приложений. К этим понятиям мы отнесем адресацию портов ввода-вывода как ячеек памяти, мультиплексирование шин и переключение банков памяти.

Адресация портов ввода-вывода как ячеек памяти

Многие микропроцессоры, включая наш иллюстративный, для выполнения обновлений информацией с портами ввода-вывода используют команды и линии управления, отличные от тех, с помощью которых осуществляются обмены с памятью. Это позволяет иметь адресацию портов, независимую от адресации ячеек памяти. Однако часто удобнее трактовать порты ввода-вывода как ячейки памяти, т. е. присвоить портам ввода-вывода адреса и обращаться к ним, как к обычным ячейкам памяти. Обычно это называют **отображением портов ввода-вывода на память**. Отображение портов ввода-вывода на память можно организовать в любом микропроцессоре, но в некоторых микропроцессорных системах этот вариант оказывается единственно возможным, поскольку специальных команд ввода-вывода, так же как и специальных управляющих линий, просто нет.

Достоинства такого отображения связаны с возможностью пользоваться при работе с портами ввода-вывода любыми командами, обращающимся к памяти. Некоторые микропроцессоры обладают богатым набором таких команд с различными способами адресации. Часто в системе команд микропроцессоров предусматриваются арифметические и логические команды

с операндами в главной памяти. Для таких микропроцессоров отображение портов ввода-вывода на память особенно выгодно.

В иллюстративном микропроцессоре мало команд, обращающихся к памяти, и тем не менее от адресации портов ввода-вывода как ячеек памяти определенные выгоды получить можно. Две специальные команды ввода-вывода обеспечивают обмен данными только между портом и аккумулятором. При отображении портов ввода-вывода на память появилась бы возможность обмениваться данными с любым из первых трех общих регистров с помощью команд загрузки регистра, запоминания регистра и пересылки. В интерфейсе с портами ввода-вывода вместо управляющих линий «ввод» и «вывод» нужно было бы использовать линии «чтение» и «запись» соответственно.

Мультиплексирование шин

Система шин в нашем микропроцессоре, как и во многих промышленных, состоит из трех отдельных шин, передающих адреса, данные и управляющую информацию. Разделение шин по трем таким функциям делает интерфейс с портами ввода-вывода и памятью простым и естественным. Тем не менее в некоторых микропроцессорах какие-то пары названных функций могут объединяться в одной шине. Как уже упоминалось в разд. 27.1, в некоторых микропроцессорах шина данных служит также для пересылки информации, управляющей передачей данных. Как правило, это делается для того, чтобы свести количество контактов интегральной схемы к некоторому приемлемому числу (скажем, 40).

В некоторых микропроцессорах успех достигается путем объединения функций адресной шины и шины данных. При этом возникает **мультиплексируемая шина** (ее часто называют **общей шиной адресов и данных**), по которой передаются адреса в одни моменты времени, а данные — в другие. Варианты этой схемы в основном различаются способами передачи адресов через общую шину. Если число битов в адресе равно числу битов в слове данных, что имеет место в некоторых микропроцессорах, то все биты адреса одновременно передаются через общую шину. Если же число битов в адресе превосходит число битов в слове данных, как это имеет место в нашем иллюстративном и многих других микропроцессорах, то можно либо предусмотреть дополнительные линии для лишних разрядов адреса, либо передавать адрес последовательно по частям.

В любом случае адресная информация, прошедшая через шины, должна каким-то образом быть запомнена для обращения к соответствующей ячейке памяти или порту ввода-вывода,

Этого можно добиться, подавая в соответствующие моменты от микропроцессора по специальным управляющим линиям сигналы, стробирующие прием адресной информации с шин на регистры.

Для работы с микропроцессорами, имеющими общие шины адресов и данных, выпускаются модули памяти со встроенным адресным регистром. Как правило, младшие разряды адреса, характеризующие ячейку внутри модуля, принимаются всеми модулями. Старшие же разряды адреса используются для выбора нужного модуля из нескольких входящих в систему ЗУ.

Выбор модуля можно осуществлять несколькими путями. Старшие разряды адреса могут либо поступать с дополнительных адресных линий микропроцессора, либо с центрального регистра в самой системе ЗУ, куда они могли быть занесены с общих шин ранее. Другой вариант выбора требуемого модуля заключается в использовании флажкового триггера выборки, имеющегося в каждом модуле. Каждый флажок в этом случае должен устанавливаться при появлении в соответствующий момент времени на общей шине определенной комбинации старших разрядов адреса. После того как адрес требуемой ячейки задан, передача данных в ячейку или из нее выполняется через общую шину обычным образом.

Как легко видеть, применение общей шины адресов и данных позволяет сократить число контактов в интегральной схеме микропроцессора. Более того, применение модулей памяти с адресными регистрами приводит к упрощению монтажных соединений между памятью и микропроцессором. Однако производительность микропроцессорной системы при использовании одной шины и для адресов, и для данных в общем случае должна снижаться.

Переключаемые банки памяти

Для микропроцессоров, как правило, адресуемое пространство памяти, определяемое числом битов в адресе, оказывается достаточным для большинства приложений. В тех же случаях, когда решаемая задача требует использования памяти большего объема, чем допускает система адресации, можно обратиться к идее переключаемых банков памяти. Выбор нужного банка определяется содержимым регистра, называемого указателем банка и подключаемого к микропроцессору в качестве порта вывода.

Предположим, что переключаемой должна быть только та часть памяти, в которой содержатся данные, но не команды, поскольку в противном случае переключение банка вызвало бы нарушение последовательного выполнения команд. Поэтому,

будем считать, что программа расположена в первой половине пространства, адресуемого обычным образом (в половине с младшими адресами). Переключение банков в этом случае будет затрагивать только вторую, старшую половину адресуемого пространства. Таким образом, когда старший бит в адресе равен 0, обращение будет всегда производиться к программной половине памяти, к ячейке, задаваемой остальными разрядами адреса. С другой стороны, когда старший бит адреса равен 1, обращение будет производиться к одному из нескольких банков памяти.

При такой организации программа может переключать банки, просто посылая номер нужного банка на регистр — указатель банка с помощью команды вывода.

Описанную схему переключения банков можно расширить на любое число банков. Если разрядности одного порта вывода недостаточно для нумерации всех нужных банков, то можно воспользоваться двумя или большим числом портов.

27.7. ЦИФРО-АНАЛОГОВЫЕ ПРЕОБРАЗОВАТЕЛИ

Во многих приложениях микропроцессоры должны работать с непрерывными физическими параметрами. В задачах управления механическими устройствами, например, обычно мы сталкиваемся с необходимостью опрашивать или управлять такими параметрами, как скорость, координаты в пространстве, сила. В конечном счете эти параметры представляются значениями электрических величин, которые мы будем считать напряжениями. Напряжение, которое представляет физический параметр, мы будем называть **аналоговой** величиной.

Аналоговые напряжения генерируются соответствующими датчиками (например, тахометром для скорости, датчиком давления для давления и силы, потенциометром для координаты в пространстве). При управлении параметрами их физические значения генерируются по соответствующим напряжениям с помощью различных приводов (моторов для сил, скоростей или координат, нагревателей для температур и т. п.). Микропроцессоры по самой своей природе устройства цифровые и работают с цифровым представлением величин. Чтобы микропроцессоры могли работать с аналоговыми величинами, нужны средства преобразования цифровых величин в аналоговые и обратно.

Цифро-аналоговые преобразования

Сначала рассмотрим преобразование цифровых величин в аналоговую форму. Для простоты пока будем считать их

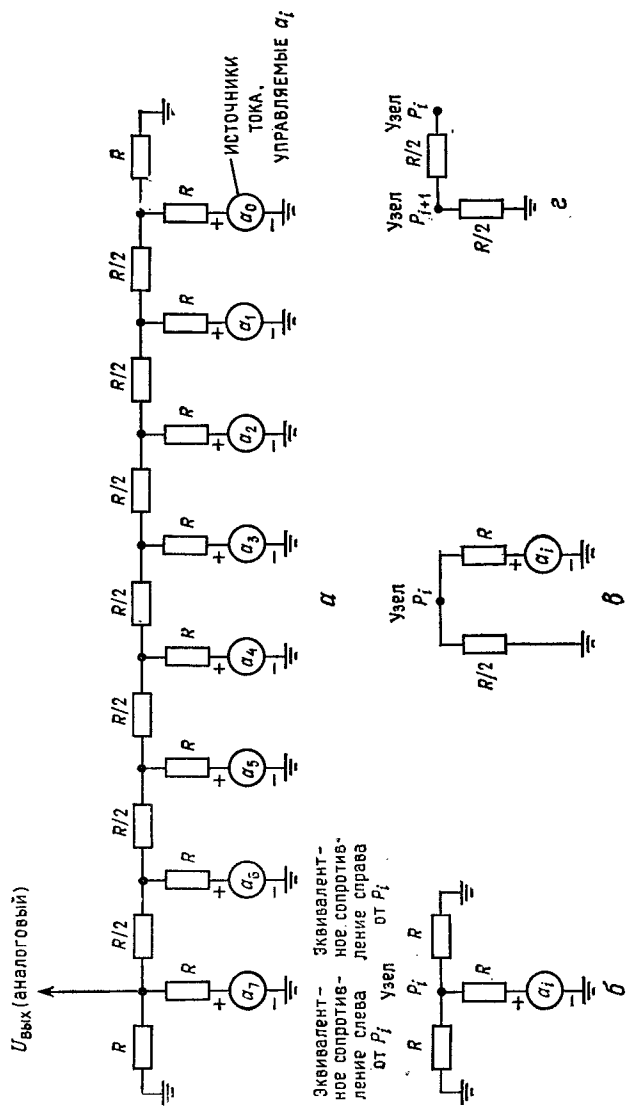


Рис. 27.18. Цифро-аналоговый преобразователь на резисторах.
 a — лестница резисторов; b — эквивалентные сопротивления левой и правой частей схемы относительно узла P_i ; c — эквивалентный делитель напряжения, определяющий действие a_i на напряжение в узле P_i ; d — эквивалентный делитель напряжения в узле P_{i+1} в зависимости от напряжения в узле P_i .

положительными. Цифровое двоичное представление величины N состоит из k -битовой цепочки $a_{k-1}a_{k-2} \dots a_1a_0$.

Значение величины определяется полиномом

$$N = a_{k-1}2^{k-1} + a_{k-2}2^{k-2} + \dots + a_12 + a_0.$$

Вычисление полинома сводится к домножению отдельных битов на весовые коэффициенты, являющиеся степенями двойки, и в сложении получаемых членов. Это соответствует вычислению линейной комбинации битов. Подобные линейные комбинации легко реализуются схемами из резисторов. Если мы предполагаем напряжениями, представляющими значения отдельных битов, то с помощью схемы из резисторов мы можем получить напряжение, соответствующее значению всего двоичного числа.

Одна из таких схем, по внешнему виду напоминающая лестницу (в технической литературе называемая аттенуатором), представлена на рис. 27.18,а. Число битов предполагается равным 8. К схеме подключаются 8 идеальных генераторов напряжения, соответствующих отдельным битам a_i . Каждый генератор дает напряжение либо 0 В, либо 1 В в зависимости от значения соответствующего бита. Таким образом, значение напряжения на генераторе численно равно значению бита. Выходное напряжение $U_{\text{вых}}$, как будет показано, пропорционально сумме значений напряжений на генераторах, взвешенных по степеням 2. Таким образом, «лестница» резисторов выполняет функции цифро-аналогового преобразователя числа N .

Вывод формулы для резисторной «лестницы»

Чтобы показать, что схема на рис. 27.18,а действительно реализует суммирование с весами, воспользуемся принципом суперпозиции линейных схем. Каждый генератор является источником входного сигнала для линейной схемы, состоящей только из резисторов. Принцип суперпозиции позволяет нам вычислить выходное напряжение $U_{\text{вых}}$, суммируя воздействия каждого индивидуального входного напряжения. Каждое из этих воздействий можно получить, вычислив $U_{\text{вых}}$ в предположении, что подано только одно входное напряжение, а все остальные входные напряжения равны 0.

Рассмотрим воздействие на выход напряжения, соответствующего биту a_i . Генератор напряжения подключен к резистору с сопротивлением R . Обозначим точку на противоположном конце этого резистора через P_i . Сопротивление левой части схемы между точкой P_i и землей оказывается равным R , как показано на рис. 27.18,б. Это легко доказать, если вычислять сопротивление, продвигаясь от левого конца схемы направо.

Крайний левый резистор с сопротивлением R включен параллельно с резистором R , идущим к генератору (поскольку на генераторе нулевое напряжение, резистор, подключенный к

нему, можно считать заземленным). Параллельно включенные резисторы обладают общим сопротивлением $R/2$, которое последовательно с расположенным правее резистором $R/2$ дает сопротивление на землю, равное R . Повторяя этот процесс, мы дойдем до узла P_i и убедимся в справедливости сделанного утверждения. Точно так же легко видеть, что сопротивление правой части схемы от узла P_i на землю равно R . Таким образом, эквивалентную схему можно представить в виде, показанном на рис. 27.18, б.

Теперь можно определить напряжение в точке P_i . Сопротивления правой и левой части схемы включены в параллель и дают общее сопротивление $R/2$, как показано на рис. 27.18, в. Это сопротивление и сопротивление R резистора, подключенного к генератору a_i , образуют делитель напряжения. Благодаря этому делителю к точке P_i оказывается приложенным часть напряжения a_i :

$$\text{Напряжение в узле } P_i = \frac{R/2}{R + R/2} a_i = \frac{1}{3} a_i.$$

Теперь, продвигаясь от узла к узлу налево, мы можем определить, чему равно выходное напряжение схемы, если в узле P_i напряжение равно $1/3 a_i$.

Очевидно, напряжение на соседнем узле P_{i+1} равно половине напряжения в узле P_i . Действительно, сопротивление между узлом P_{i+1} и землей, если не считать резистор, соединяющий узлы P_i и P_{i+1} , равно $R/2$. Это значение сопротивления дают два параллельно включенных сопротивления, а именно сопротивление левой части схемы от узла P_{i+1} и сопротивление резистора к генератору a_i . Сопротивление $R/2$ между P_{i+1} и землей вместе с сопротивлением $R/2$ резистора между P_i и P_{i+1} образует делитель напряжения, показанный на рис. 27.18, г. Этот делитель дает в узле P_{i+1} напряжение, равное половине от напряжения в узле P_i , т. е.

$$\text{Напряжение в узле } P_{i+1} = \frac{R/2}{R/2 + R/2} \times \text{Напряжение } P_i = \frac{1}{2} \frac{a_i}{3}.$$

Напряжение уменьшается вдвое при каждом шаге влево от узла к узлу. Поэтому очевидно, что выходное напряжение в узле P_7 задается формулой

$$\text{Напряжение в узле } P_7 = \left(\frac{1}{2}\right)^{7-i} \frac{a_i}{3}.$$

Суммируя воздействие от всех генераторов, мы получим результирующее выходное напряжение

$$U_{\text{вых}} = \sum_{i=0}^7 \left(\frac{1}{2}\right)^{7-i} \frac{a_i}{3} = \left(\frac{1}{2}\right)^7 \frac{1}{3} \sum_{i=0}^7 \left(\frac{1}{2}\right)^{-i} a_i = \left(\frac{1}{2}\right)^7 \frac{1}{3} \sum_{i=0}^7 2^i a_i.$$

Сумма в полученном выражении представляет собой полином, определяющий значение N . Таким образом, $U_{\text{вых}}$ пропорционально N . Коэффициент пропорциональности равен $(1/2)^7(1/3)$ или в общем случае $(1/2)^{k-1}(1/3)$.

Преобразование чисел, которые могут принимать как положительные, так и отрицательные значения и представлены в дополнительном коде, можно получить, несколько видоизменив схему для положительных чисел. Представление чисел в дополнительном коде можно считать аналогичным двоичному представлению положительных чисел, только с отрицательным весом у старшего (знакового) бита. Таким образом, число N , представленное k -битовым дополнительным кодом $a_{k-1}a_{k-2} \dots a_1a_0$, определяется полиномом

$$N = a_{k-1}(-2^{k-1}) + a_{k-2}2^{k-2} + \dots + a_12^1 + a_0.$$

Преобразователь в аналоговую форму для этого выражения можно реализовать такой же «лестницей» резисторов, где полярность генератора напряжения a_{k-1} изменена на противоположную.

Реализация генераторов напряжения

Теперь остановимся на реализации генераторов напряжения, дающих входные сигналы на «лестницу» резисторов. По значению бита a_i они должны выдавать напряжение, равное или пропорциональное a_i . Важно, чтобы генераторы приближались к идеальным, т. е. чтобы их выходные напряжения не зависели от нагрузки со стороны схемы резисторов. Также важна точность выходных напряжений, поскольку она непосредственно влияет на точность всего преобразователя.

Эти требования можно удовлетворить с помощью аналоговых переключателей, которые соединяют каждую входную линию «лестницы» либо с высокоточным источником эталонного напряжения, либо с землей в зависимости от значения a_i . В качестве такого переключателя можно воспользоваться схемой на биполярных транзисторах с активной нагрузкой, см. рис. 27.19. При этом предполагается, что уровень напряжения, соответствующий логической 1, чуть больше, чем $U_{\text{эталон}}$. Если a_i равно логической 1, верхний транзистор будет в состоянии насыщения, в результате чего на выходе схемы будет напряжение, равное эталонному. Если a_i равно логическому 0, то в насыщении будет нижний транзистор, и на выходе будет потенциал земли. Транзисторы должны выбираться с пренебрежимо малым падением напряжения между коллектором и эмиттером в состоянии насыщения.

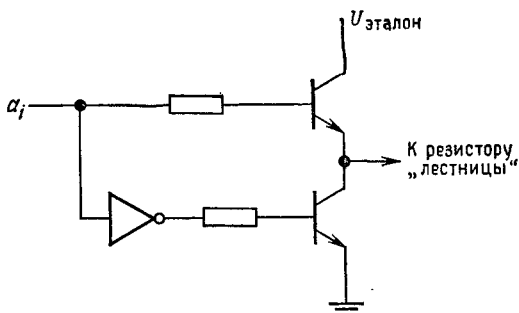


Рис. 27.19. Схема генератора напряжения, управляемого на двух биполярных транзисторах.

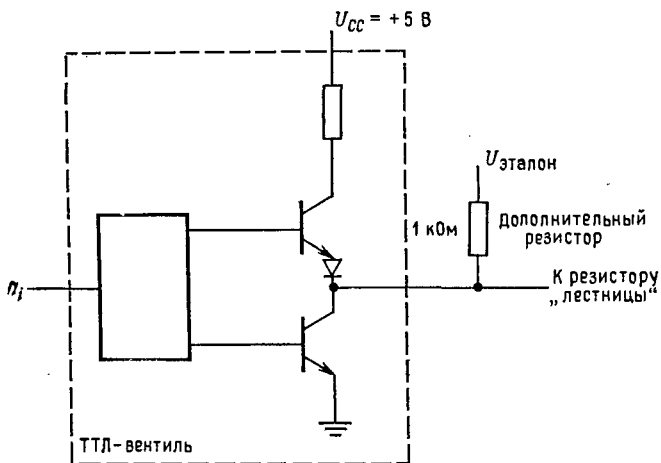


Рис. 27.20. Схема генератора напряжения на ТТЛ вентиле с дополнительным резистором.

При небольшом числе разрядов (не более восьми) приемлемые результаты можно получить на ТТЛ-вентиле с активной нагрузкой в выходном каскаде и дополнительным резистором, как показано на рис. 27.20. Обычно на части схемы в ТТЛ-вентиле, обеспечивающей высокий выходной потенциал, имеется заметное падение напряжения. Дополнительный резистор между выходом вентиле и эталонным напряжением (номинально равным $+5\text{В}$) приближает выходное напряжение к эталонному при условии, что нагрузка со стороны резисторов «лестницы» достаточно мала. Сопротивление дополнительного резистора на схеме выбрано равным 1 кОм , где-то близко к минимальному значению, при котором нижний транзистор не

перегружается. Входные резисторы в «лестнице» нужно выбирать с существенно большими значениями сопротивлений. Резисторы порядка 20 кОм обычно оказываются приемлемыми.

Преобразователь на операционном усилителе

Вместо лестницы резисторов для суммирования с весами в соответствии с представлением двоичного числа в виде полинома можно использовать **операционный усилитель**. Операционным называют усилитель напряжения с большим отрицательным коэффициентом усиления ($-A$), который можно запрограммировать для выполнения различных операций. Программирование осуществляется подключением резисторов или других пассивных элементов. На рис. 27.21 показана схема, в которой операционный усилитель запрограммирован для выполнения цифро-аналогового преобразования.

Сигналы U_i — это входные напряжения, соответствующие значениям битов a_i в преобразуемом двоичном числе N . Выходное напряжение схемы $U_{\text{вых}}$ пропорционально значению числа N . Чтобы доказать это, просуммируем токи, проходящие через точку схемы, обозначенную как «общая точка». Обозначим напряжение в этой точке через U . Каждый входной источник напряжения соединен с общей точкой через резистор, который мы для общности обозначим через R_i . По закону Ома ток через каждый из этих резисторов равен $(U_i - U)/R_i$. Выходная линия, подключенная к общей точке через резистор R , вносит свою долю тока, равную $(U_{\text{вых}} - U)/R$. Сумма этих токов должна равняться 0, т. е.

$$\frac{U_0 - U}{R_0} + \frac{U_1 - U}{R_1} + \dots + \frac{U_7 - U}{R_7} + \frac{U_{\text{вых}} - U}{R} = 0.$$

Далее, поскольку напряжение U подано на вход операционного усилителя, его выходное напряжение $U_{\text{вых}}$ должно равняться $-AU$, где A — коэффициент усиления. Таким образом, $U = -U_{\text{вых}}/A$. Подставив U в предыдущее уравнение и собрав члены с $U_{\text{вых}}$, мы получим

$$\begin{aligned} & \frac{U_0}{R_0} + \frac{U_1}{R_1} + \dots + \frac{U_7}{R_7} = \\ & = -\frac{U_{\text{вых}}}{A} \left(\frac{1}{R_0} + \frac{1}{R_1} + \dots + \frac{1}{R_7} + \frac{A}{R} + \frac{1}{R} \right). \end{aligned}$$

Это уравнение можно упростить, проведя аппроксимацию, опирающуюся на высокое значение коэффициента усиления операционного усилителя. Обычно это значение бывает не менее 50 000, из чего следует, что член A/R в правой части уравнения много больше всех остальных членов, входящих в сумму.

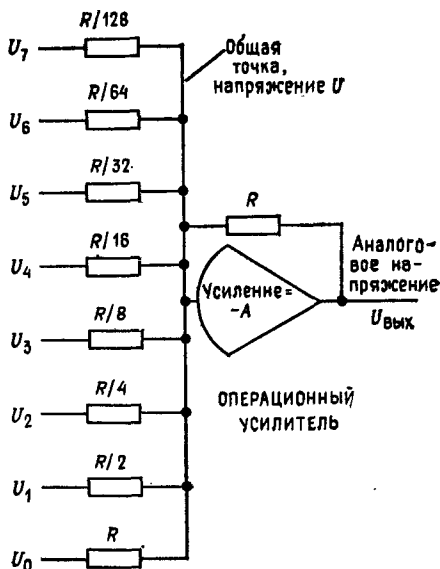


Рис. 27.21. Двоичный цифро-аналоговый преобразователь с операционным усилителем.

Поэтому величина выражения в скобках в правой части с хорошей точностью равна A/R . Вся правая часть при этом становится равной

$$\frac{-U_{\text{вых}}}{A} \left(\frac{A}{R} \right) = \frac{-U_{\text{вых}}}{R}.$$

Следовательно, выходное напряжение $U_{\text{вых}}$ определяется выражением

$$[U_{\text{вых}} = -R \left(\frac{U_0}{R_0} + \frac{U_1}{R_1} + \dots + \frac{U_7}{R_7} \right).$$

Если теперь подставить значения R_i , указанные на рис. 27.20, мы получим

$$\begin{aligned} U_{\text{вых}} &= -R \left(\frac{U_0}{R} + \frac{U_1}{R/2} + \dots + \frac{U_7}{R/128} \right) = \\ &= -(U_0 + 2U_1 + \dots + 128U_7) = -\sum_{i=0}^7 U_i 2^i. \end{aligned}$$

Таким образом, значение выходного напряжения равно взятому с обратным знаком значению исходного двоичного числа.

Аналогичные схемы можно применять для преобразования в аналоговую форму чисел, представленных в коде 8421 BCD

(двоично-кодированных десятичных), приведя только значения входных сопротивлений в соответствие с весами битов в коде BCD. Двухразрядное десятичное число N , представляемое кодом BCD

$$N = b_3 b_2 b_1 b_0 a_3 a_2 a_1 a_0,$$

имеет значение

$$N = (b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0) 10^1 + (a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0).$$

Следовательно, веса битов имеют значения: 80, 40, 20, 10, 8, 4, 2, 1. Поэтому входные резисторы схемы, аналогичной рис. 27.21, должны иметь следующие значения:

$$\frac{R}{80} \quad \frac{R}{40} \quad \frac{R}{20} \quad \frac{R}{10} \quad \frac{R}{8} \quad \frac{R}{4} \quad \frac{R}{2} \quad R.$$

Аналого-цифровые преобразования

Мы рассмотрим несколько методов преобразования аналоговых величин в цифровую форму, использующих в режиме «проб и ошибок» цифро-аналоговые (ЦА) преобразователи. При любом способе все начинается с некоторого исходного испытуемого числа, которое преобразуется цифро-аналоговым преобразователем в аналоговую форму и сравнивается с заданным аналоговым напряжением. По результатам сравнения испытуемое число корректируется. Затем попытка повторяется с новым числом и т. д., пока не будет подобрано число, соответствующее заданному напряжению. При этом предполагается, что входное аналоговое напряжение промасштабировано и соответствует диапазону ЦА-преобразователя.

Простейший метод подбора — это начать с наименьшего числа в диапазоне (с нуля при положительных числах) и последовательно увеличивать его на 1 до тех пор, пока преобразованное напряжение не превысит входное или не сравняется с ним. На рис. 27.22 показано, как это можно реализовать с помощью суммирующего счетчика. Выходы счетчика поданы на ЦА-преобразователь. Аналоговый выход этого преобразователя и заданное входное напряжение поданы на компаратор (схему сравнения), на выходе которого появляется логическая 1, если заданное напряжение превышает полученное ЦА-преобразователем. Выход от компаратора и сигналы от генератора импульсов подаются на вентиль И, выход последнего подается на счетный вход счетчика.

Процесс аналого-цифрового (АЦ) преобразования начинается со сброса счетчика в нуль. После этого счетчик начинает увеличиваться, пока на выходе компаратора не появится логический 0. Это произойдет, когда содержимое счетчика окажется

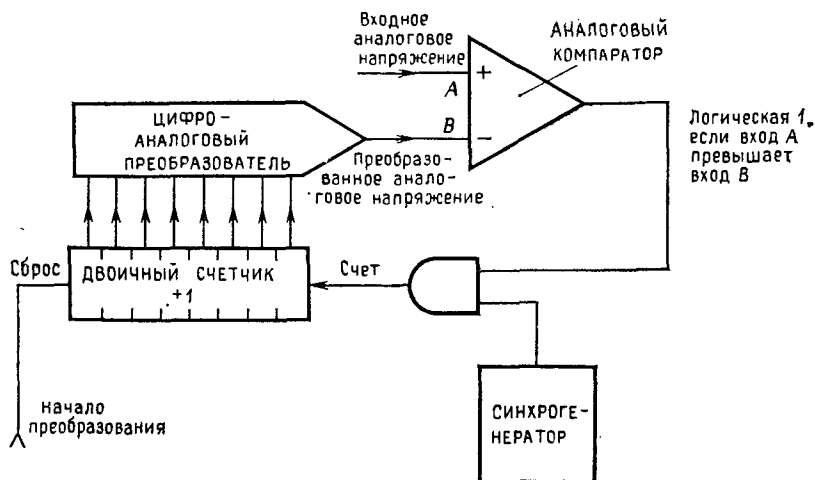


Рис. 27.22. Аналого-цифровой преобразователь с суммирующим счетчиком.

достаточным для того, чтобы выход ЦА-преобразователя превысил или сравнялся с входным напряжением. Достигнутое к этому моменту содержимое счетчика и берется в качестве выходного преобразованного числа.

Можно существенно улучшить описанный метод АЦ-преобразования, если суммирующий счетчик заменить реверсивным. Тогда для корректировки испытуемого числа можно либо увеличивать, либо уменьшать счетчик. В этом случае появляется возможность непрерывно отслеживать изменяющееся входное напряжение. Схема такого преобразователя показана на рис. 27.23.

Для сравнения заданного аналогового напряжения с выходом ЦА-преобразователя используется компаратор с двумя выходами H и L . Логическая 1 на выходе H означает, что напряжение с выхода ЦА-преобразователя превышает входное аналоговое напряжение по крайней мере на некоторую величину Δ . Логическая 1 на выходе L означает, что выход ЦА-преобразователя по крайней мере на Δ ниже входного аналогового напряжения. Таким образом, появляется «мертвая зона» шириной 2Δ , в которой ни на одном из выходов нет логической 1. Каждый из выходов управляет своим вентилем, формирующим сигналы соответственно увеличения и уменьшения счетчика.

Процесс преобразования может начинаться с произвольного значения на счетчике. Если это значение велико по сравнению с входным аналоговым напряжением, логическая 1 на

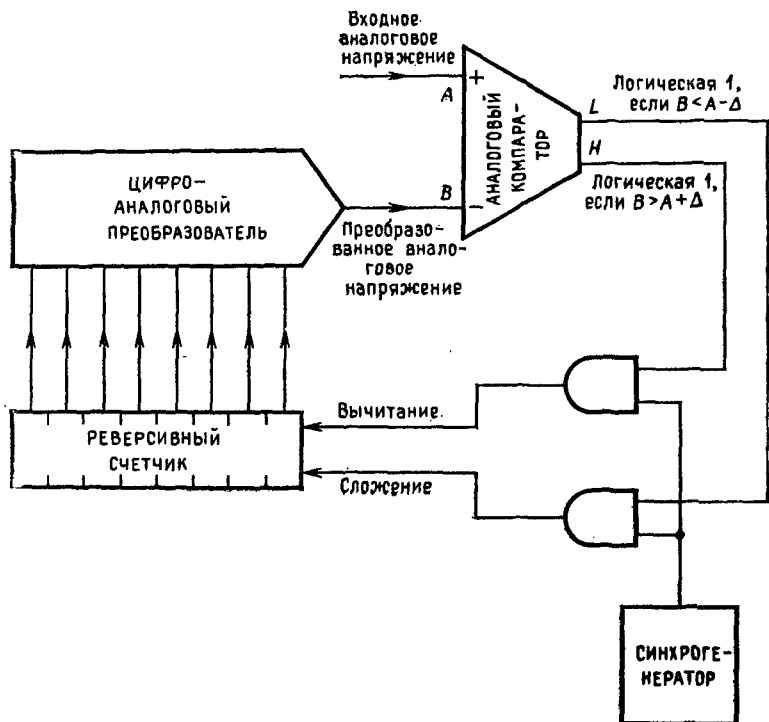


Рис. 27.23. Аналого-цифровой преобразователь с реверсивным счетчиком.

выходе H компаратора приведет к тому, что счетчик при очередном импульсе уменьшится на 1. Если же значение на счетчике мало по сравнению с входным аналоговым напряжением, логическая 1 на входе L приведет к его увеличению. Когда значение на счетчике попадет в диапазон $\pm \Delta$ относительно входного аналогового напряжения, оба выхода H и L примут нулевые значения, и счетчик перестанет изменяться. Если входное аналоговое напряжение изменится после этого, то соответственно изменится и значение на счетчике.

Значение Δ должно выбираться таким образом, чтобы в мертвую зону попадало только одно значение счетчика. Ширина мертвой зоны 2Δ должна быть больше изменения аналогового напряжения на выходе ЦА-преобразователя при переходе к соседнему числу, что соответствует аналоговому весу младшего бита счетчика ($W_{\text{МБ}}$). Однако ширина мертвой зоны не должна быть больше двух весов младшего бита счетчика

$$W_{\text{МБ}} < 2\Delta < 2W_{\text{МБ}}.$$

Время срабатывания преобразователя с реверсивным счетчиком зависит от того, насколько далеки друг от друга начальное и результирующее значения. Число импульсов счета может варьироваться от 0 до полного диапазона счетчика. В тех случаях, когда преобразуется медленно меняющаяся величина, преобразования, кроме самого первого, завершаются за несколько импульсов.

Метод последовательных приближений

Во многих случаях, однако, входное аналоговое напряжение от одного преобразования к следующему может сильно изменяться. Это в особенности верно в тех довольно частых случаях, когда один АЦ-преобразователь используется для последовательного преобразования аналоговых сигналов от многих датчиков (режим мультиплексирования). Более того, в некоторых приложениях требуется осуществлять АЦ-преобразования со строгой периодичностью за фиксированное время, не зависящее от конкретных значений сигнала.

Метод АЦ-преобразования, выполняемого за фиксированное и относительно небольшое время, существует. Согласно этому методу, назовем его **методом последовательных приближений**, последовательно подбираются значения битов, все ближе приближающиеся к входному аналоговому напряжению. Для преобразователей, работающих с положительными числами, преобразование начинается с числа, содержащего нули во всех битах. Затем старший бит устанавливается в 1, число преобразуется и сравнивается с входным напряжением. Если полученное число больше входного напряжения, старший бит сбрасывается в 0; в противном случае остается равным 1. Затем следующий бит, старший из оставшихся, устанавливается в 1, и полученное число снова преобразуется и сравнивается с входным напряжением. Если оказывается, что число велико, бит сбрасывается. Этот процесс повторяется для всех более младших битов, причем значения ранее подобранных старших битов не изменяются. После подбора младшего бита преобразование завершается.

Схема, реализующая описанный метод, показана на рис. 27.24. Испытуемое число хранится на регистре из RS-триггеров, переключаемых задним фронтом. Как всегда, содержимое регистра подается на вход ЦА-преобразователя. Выход ЦА-преобразователя и входное аналоговое напряжение поступают на компаратор, выход которого управляет сбросом подбираемого бита.

Для организации последовательного просмотра битов испытуемого числа слева направо используется четырехбитовый

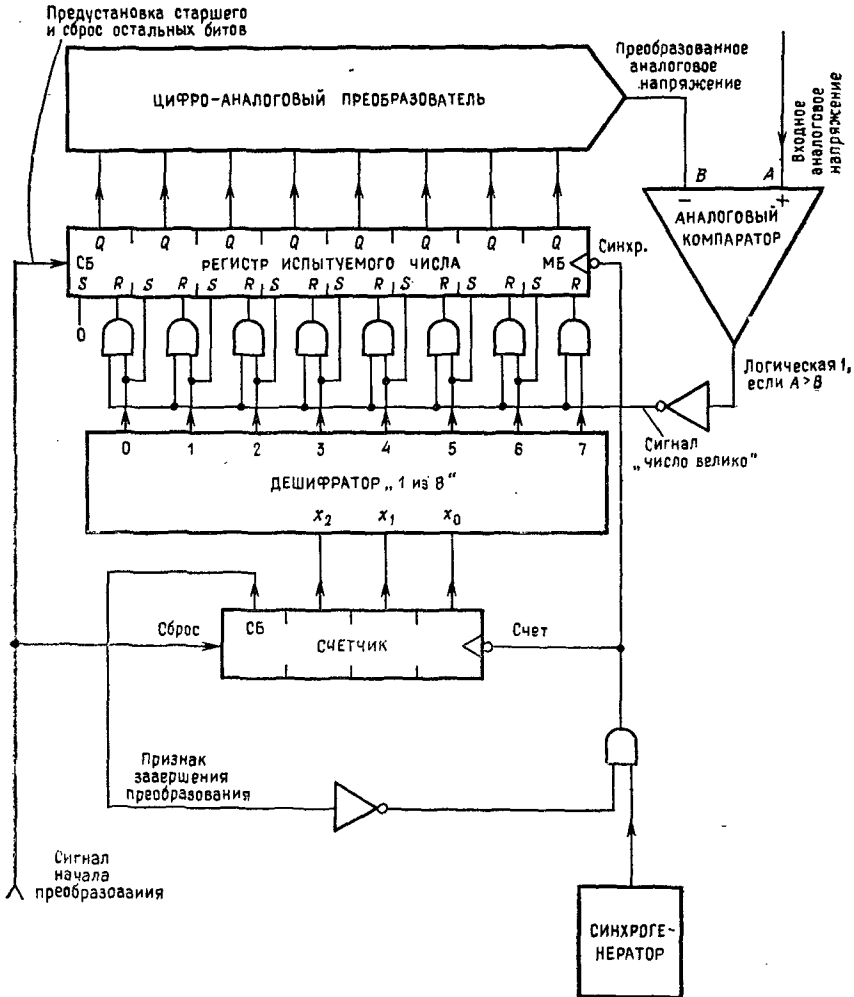


Рис. 27.24. Аналого-цифровой преобразователь, работающий по методу последовательных приближений.

счетчик и дешифратор «1 из 8». Входами дешифратора служат три младших разряда счетчика, которые определяют, какая из восьми выходных линий дешифратора содержит логическую 1. Импульсы от генератора поступают на счетный вход счетчика, а также на триггеры регистра испытываемого числа в качестве синхроимпульсов. Регистр и счетчик тем самым работают в синхронном режиме, и их состояния меняются по заднему

фронту импульсов. Старший разряд счетчика, который равен 1, когда счетчик досчитывает до 8, используется для отключения от генератора и, следовательно, остановки процесса преобразования после 8 импульсов.

С помощью выходных линий дешифратора осуществляется выбор соответствующих входов на триггерах регистра для сброса подбираемого разряда и установки следующего. Сигнал сброса при этом зависит от выхода компаратора. Таким образом, когда приходит очередной импульс, триггер, соответствующий активной линии дешифратора, сбрасывается, если число велико, а следующий триггер устанавливается в 1 независимо от результата сравнения.

Схема в целом работает следующим образом. Импульс на линии «начало преобразования», который предполагается синхронизированным с импульсами генератора, устанавливает в качестве начального значения испытываемого числа 1 в старшем бите и нули в остальных. Этот же начальный импульс сбрасывает в 0 четырехбитовый счетчик и тем самым открывает импульсы от генератора. Начальное число преобразуется в аналоговую форму и сравнивается с входным напряжением. По заднему фронту импульса от генератора старший бит испытываемого числа сбрасывается, если число велико. В любом случае следующий бит будет установлен в 1. Кроме того, счетчик будет увеличен на 1, и процесс повторится для следующего бита. Когда счетчик досчитает до 8, все биты в числе будут определены. Процесс остановится, и на регистре будет результирующее число.

В общем случае метод последовательных приближений требует k импульсов для преобразования аналогового напряжения в k -битовое число, тогда как метод с реверсивным счетчиком в худшем случае может потребовать $2^k - 1$ импульсов.

Аналого-цифровое преобразование с помощью микропроцессора

Если аналоговая величина преобразуется в цифровую форму для использования в микропроцессоре, то логично и выгодно, чтобы микропроцессор взял на себя работу по преобразованию. За счет этого можно упростить внешние схемы. Более того, это может упростить и связи между микропроцессором и внешними схемами.

Участие микропроцессора в процессе преобразования оказывается особенно эффективным, если преобразовывать нужно много сигналов. В этом случае испытываемые числа для всех аналоговых сигналов можно преобразовывать с помощью единственного ЦА-преобразователя, подключенного к выходному

порту. Сравнение выхода ЦА-преобразователя с каждым входным сигналом выполняет отдельный компаратор. Результат каждого сравнения поступает в микропроцессор по отдельной линии в порт ввода.

Для большей наглядности рассмотрим схему преобразования сигналов, поступающих по восьми аналоговым каналам, представленную на рис. 27.25. На схеме мы видим один вводной и один выводной порт с общим кодом устройства. Регистр выводного порта принимает с шин данных испытуемое число и подает его на входы ЦА-преобразователя, выход которого подключен к входам восьми компараторов. На второй вход каждого компаратора подается сигнал по одному из восьми аналоговых входных каналов. Выходы компараторов поданы на тристабильные формирователи, образующие порт ввода.

Описанных внешних компонент и связей достаточно для того, чтобы можно было программно выполнить аналого-цифровое преобразование для всех восьми каналов. В общем случае программа начинает преобразование для некоторого канала, выводя соответствующее испытуемое число. Это число преобразуется в аналоговое напряжение и сравнивается с входными напряжениями во всех каналах.

Результаты всех восьми сравнений программа получает через порт ввода. Она выделяет и тестирует бит, который соответствует интересующему ее каналу и определяет новое пробное значение числа. Подобрав число для напряжения в одном канале, микропроцессор переходит к другому каналу и т. д. Таким образом, преобразование по всем каналам ведется в режиме временного мультиплексирования. Следует отметить, что между микропроцессором и внешними схемами не потребовалось передавать информацию о состоянии. Это объясняется тем, что преобразование выполняет микропроцессор, и он контролирует как начало, так и конец преобразования. Если же преобразование выполняется полностью внешними компонентами, то для информации о состоянии нужно выделять специальные порты.

Метод преобразования, используемый в программе, может быть любым, включая метод последовательных приближений, методы со счетчиками и другие.

Существует интересный программный метод, который оказывается довольно эффективным, когда число битов невелико (скажем, 4), а число каналов относительно велико (скажем, 16 или больше). Согласно этому методу, пробуются последовательно все числа диапазона, начиная с 0. Для каждого числа регистрируются результаты всех сравнений. Если при пробе очередного числа компаратор некоторого канала изменил свой выход, то это число и берется в качестве результата для этого

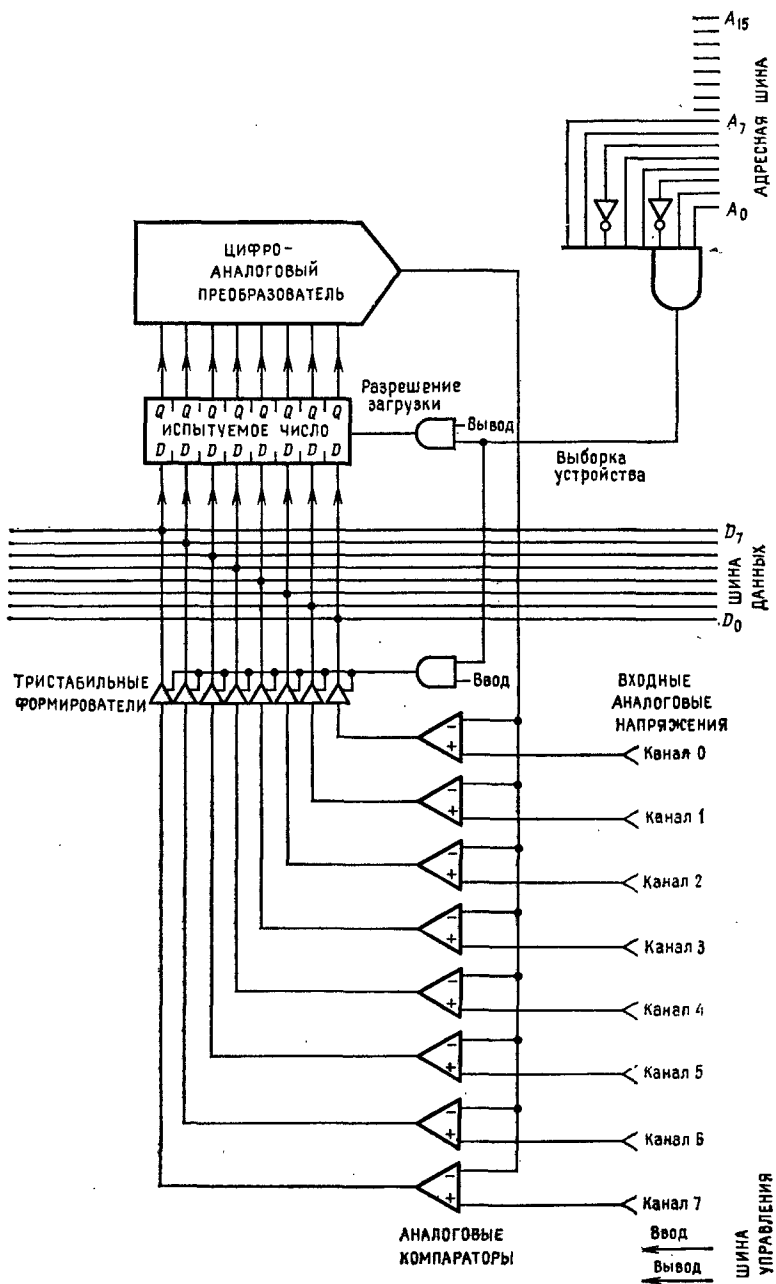


Рис. 27.25. Интерфейс иллюстративного микропроцессора для выполнения АЦ-преобразования аналоговых сигналов, поступающих по восьми каналам,

канала. Перебрав все числа, мы завершим преобразование для всех каналов. При 4 бит и 16 каналах преобразование завершится через 16 попыток, т. е. в среднем будет затрачена одна попытка на канал.

27.8. ПОСЛЕДОВАТЕЛЬНЫЙ ВВОД-ВЫВОД

Передача данных между микропроцессором и устройствами ввода-вывода (например, терминалами) или внешними запоминающими устройствами (например, кассетным магнитофоном) часто осуществляется в последовательном режиме. В общем случае передача в последовательном режиме происходит по одной линии, причем биты, составляющие слово, передаются друг за другом.

Оба случая, с устройствами ввода-вывода и с внешними ЗУ, можно рассмотреть вместе. В каждом из них есть источник информации и получатель. Данные поступают от источника и затем либо передаются бит за битом через линию получателю, либо накапливаются бит за битом в некотором устройстве для последующего использования получателем. Путь, по которому передаются данные от источника к получателю, в обоих случаях называется **каналом**.

Часто оказывается, что на одном или обоих концах последовательного канала данные нужны в параллельной форме. В этом случае нужно предусмотреть средства преобразования данных из одной формы в другую. Естественное решение этой проблемы дают сдвиговые регистры. Для преобразования из параллельной формы в последовательную данные сначала параллельно загружаются в сдвиговый регистр. После этого они многократно сдвигаются в одну или другую сторону, в результате чего на выходе одного из концевых триггеров последовательно появляются значения всех битов. Для обратного преобразования линия, в которой появляются последовательные данные, подключается к одному из концов сдвигового регистра, и сдвиговый регистр сдвигается столько раз, сколько битов в слове данных. После этого все биты слова можно получить в параллельной форме на выходах триггеров регистра.

Различают два основных способа передачи данных через последовательный канал: **синхронный** и **асинхронный**. При синхронной передаче данных должна существовать общая для источника и получателя линия синхросигналов, импульсы на которой служат для выделения индивидуальных битов в канале. Как источник, так и получатель интерпретируют один и тот же синхроимпульс как указание на наличие в канале очередного бита. Поэтому для работы с синхронным последовательным каналом не требуется точно привязываться к абсолютному

времени. При асинхронном способе источник и получатель не имеют общей синхроинии. В этом случае они должны точно отмерять время появления каждого бита относительно начала слова. Для этого источник должен пометать начало слова так, чтобы получатель мог его распознать.

Синхронная передача

Чтобы проиллюстрировать работу синхронного канала, рассмотрим схему на рис. 27.26. Схема рассчитана на слова из восьми битов, причем один бит — бит контроля по четности.

В источнике предусмотрен сдвиговый регистр для преобразования параллельных данных в последовательную форму. Параллельный прием данных на регистр строится сигналом в линии «загрузка». Семь битов вводимой информации поступают на правые семь триггеров регистра. Генератор четности по этим семи информационным битам формирует значение контрольного бита и подает его на вход восьмого триггера в регистре.

Генератор импульсов синхронизирует работу схем как на стороне источника, так и на стороне получателя. На стороне источника предусмотрен трехбитовый счетчик, ведущий счет импульсов синхронизации по модулю 8. Состояние 7 на счетчике вызывает появление стробирующего сигнала «загрузка» по очередному синхроимпульсу. По этому сигналу происходит загрузка сдвигового регистра источника очередным словом, представленным в параллельной форме на его входах. Во всех состояниях, отличных от 7, синхроимпульсы поступают на линию «сдвиг» сдвигового регистра. Таким образом происходит последовательная выдача восьми битов в линию данных канала. Самый правый бит попадает в канал сразу после загрузки сдвигового регистра при нулевом состоянии счетчика. Левый бит подается в канал после седьмого импульса сдвига.

Получатель также имеет восьмибитовый сдвиговый регистр для приема последовательных данных из канала и преобразования их в параллельную форму. Данные из канала поступают на левый вход сдвигового регистра. Синхроимпульсы, посылаемые источником по второй линии канала, подаются в сдвиговый регистр получателя на линию «сдвиг». Таким образом, данные «вдвигаются» в регистр с левого конца.

Так же как и на стороне источника, на стороне получателя есть трехбитовый счетчик, считающий синхроимпульсы по модулю 8. Предполагается, что счетчики с обеих сторон в некоторый момент были одновременно сброшены и, следовательно, считают согласованно. Каждый раз, когда счетчик получателя досчитывает до нуля, формируется сигнал «слово получено»,

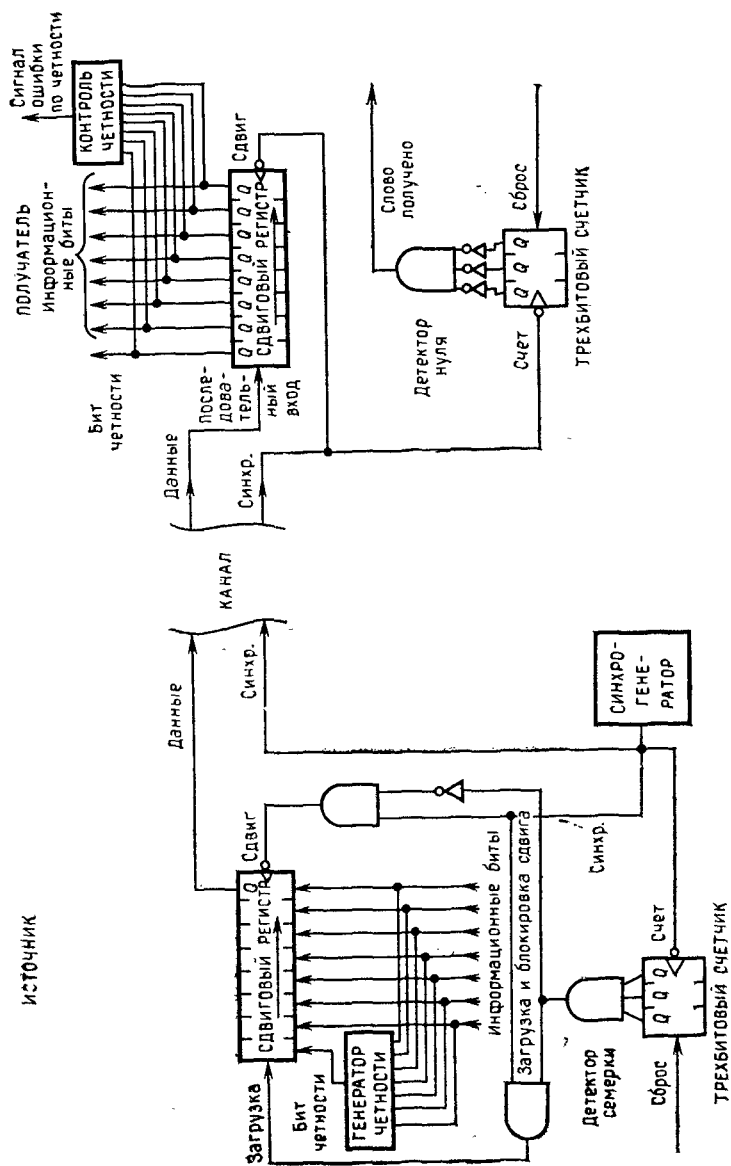


Рис. 27.26. Синхронный последовательный канал передачи данных.

означающий, что очередное слово находится на выходах триггеров сдвигового регистра. Схема контроля четности в получателе формирует сигнал ошибки, если получено слово с неправильной четностью.

Канал, связывающий источник с получателем, может принадлежать к одному из многих типов коммуникационных связей или носителей информации. Например, он может представлять собой две линии, связывающие порт ввода-вывода компьютера с периферийным устройством, может оказаться сложной системой, использующей модуляторы и демодуляторы, в случае телефонного канала или радиоканала, или это может быть пара дорожек на кассетной магнитной ленте. Важно, что задержки, которые могут возникать в синхронном канале, одновременно и одинаково влияют и на данные, и на синхросигналы, сохраняя их относительные временные соотношения.

Представленная на рис. 27.26 частная схема обладает синхронностью как в отношении отдельных битов, так и в отношении самих слов. Это действительно так, поскольку начало каждого слова строго привязано к каждому восьмому синхроимпульсу. Возможны схемы, синхронные в отношении битов и асинхронные в отношении слов. В таких случаях начало слов синхронизируется внешними импульсами.

Асинхронная передача

При асинхронной передаче данных канал работает асинхронно как в отношении битов, так и в отношении слов. Это означает, что общая синхролиния между источником и получателем, выделяющая моменты передачи отдельных битов, отсутствует, и передача слова может начаться в произвольный момент времени. Тем не менее начало слова должно быть каким-то образом отмечено, чтобы получатель мог определить момент появления каждого бита в слове.

Стандартное решение проблемы заключается в обрамлении каждого слова стартовым и стоповым битами. Пока по каналу нет передачи слов, он находится в некотором состоянии, скажем в состоянии логической 1. Стоповому биту, завершающему слово, соответствует такое же значение 1. Тогда стартовому биту, начинающему слово, соответствует противоположное, нулевое значение. Таким образом передний фронт стартового бита всегда вызывает переход состояния из 1 в 0 и может быть распознан получателем. Передний фронт стартового бита используется в получателе и как признак начала слова, и как начало отсчета времени для определения моментов прихода отдельных битов в слове.

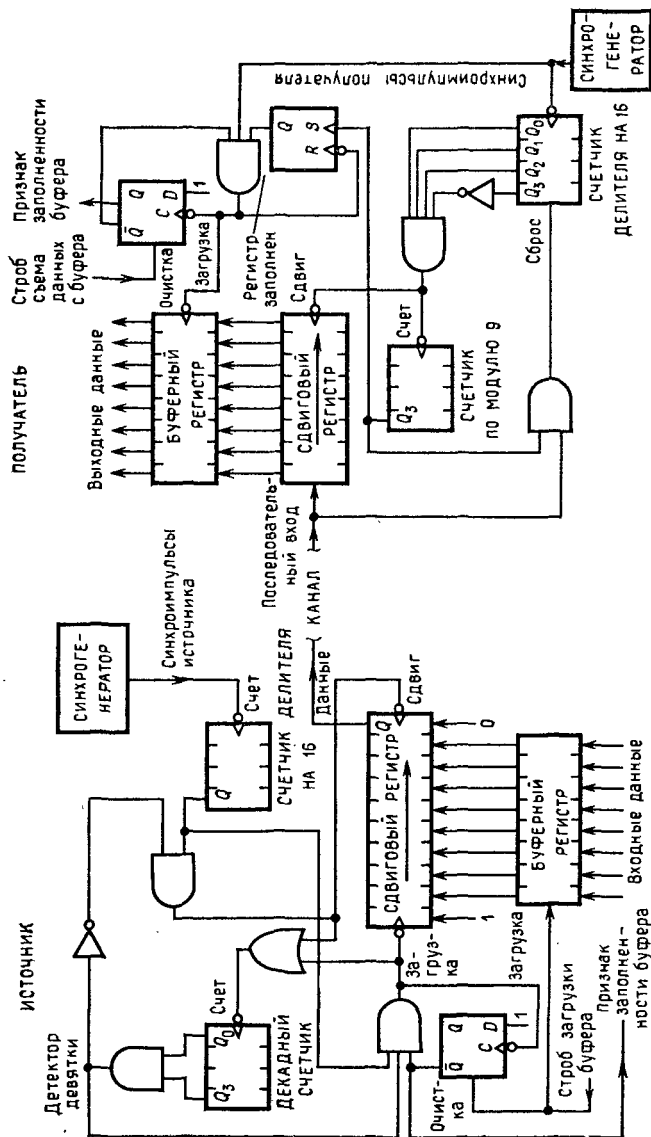


Рис. 27.27. Асинхронный последовательный канал передачи данных.

Схема, реализующая передачу по асинхронному последовательному каналу, приведена на рис. 27.27. Она рассчитана на обработку слов, имеющих восемь информационных битов. Для упрощения схемы на ней не показаны компоненты контроля по четности. Как и в случае синхронной передачи, на обеих сторонах имеется по сдвиговому регистру для преобразования данных из параллельной в последовательную форму и обратно. Сдвиговой регистр источника десятибитовый; помимо восьми информационных битов, он вмещает стартовый и стоповый биты.

Информационные биты поступают на сдвиговой регистр параллельно с буферного регистра. Буферный регистр в свою очередь получает их извне. Загрузка данных в буферный регистр стробируется внешним сигналом «строб загрузки буфера». Загрузка буфера может происходить в то время, когда предыдущее слово сдвигается в сдвиговом регистре. В этом случае передача слова из буфера в сдвиговой регистр произойдет сразу, как только последний освободится. Если же загрузка данных в буфер произойдет в момент, когда сдвиговой регистр свободен, эти данные будут переданы в сдвиговой регистр ближайшим же синхроимпульсом. В качестве флага используется *D*-триггер, хранящий признак того, что буфер не содержит данных, готовых к передаче. Этот триггер сбрасывается при загрузке буфера и устанавливается в 1 (синхроимпульсом при $D = 1$) в момент передачи содержимого буфера на сдвиговой регистр.

Импульсы, синхронизирующие работу схем источника, получают путем деления частоты задающего генератора на 16. Как мы увидим в дальнейшем, это позволяет согласовать работу источника и получателя.

Число импульсов сдвига в источнике управляется декадным счетчиком. Каждый импульс сдвига подается также и на его счетный вход. Состояние 9, соответствующее концу передачи слова, детектируется вентилем И. Выходной сигнал детектора девятки блокирует последующие импульсы сдвига и счета. В этом состоянии схема бездействует.

Когда схема бездействует, может произойти загрузка очередного слова из буфера в сдвиговой регистр. Это делается по импульсу, формирующемуся при наличии сигнала детектора девятки из заполненности буфера. Декадный счетчик продвигается этим же импульсом и переходит в состояние 0. Сдвиговой регистр начинает «выдвигать» в канал новое слово.

На стороне получателя биты принимаются на восьмибитовый сдвиговой регистр. Импульс сдвига подается примерно в середине интервала передачи бита. Импульс сдвига поступает и на сдвиговой регистр, и на счетчик по модулю 9. Состояние

этого счетчика перед началом очередного слова равно 8. Первый же сдвиговый импульс переведет его в нуль, после чего он в конце слова снова досчитает до 8. Таким образом, «сдвинутыми» в сдвиговый регистр будут 9 бит, причем стартовый бит, пройдя через весь регистр, будет потерян на правом конце.

Импульсы сдвига в приемнике формируются на счетчике делителя частоты задающего генератора на 16. В период бездействия канала этот счетчик поддерживается в сброшенном состоянии, и, пока он сброшен, импульсов сдвига нет.

Как только по каналу приходит передний фронт стартового импульса, сигнал сброса счетчика делителя частоты принимает значение логического 0 и счетчик начинает считать. На седьмом импульсе формируется сигнал сдвига, приходящийся примерно на середину стартового бита. Через каждые 16 последующих импульсов формируется следующий импульс сдвига, приходящийся примерно на середину соответствующего информационного бита.

Следует обратить внимание, что в том случае, когда после начала стартового бита, но еще до появления сдвигового импульса канал вернется в состояние 1, счетчик делителя частоты снова окажется сброшенным. Это будет означать, что за начало стартового сигнала была принята помеха в канале. Схема в этом случае снова перейдет в режим ожидания полноценного стартового сигнала.

На стороне получателя также есть буферный регистр. В него загружается содержимое сдвигового регистра, если буфер свободен, а сдвиговый регистр заполнен. Признаки заполненности этих регистров хранятся на двух триггерах. Триггер заполненности сдвигового регистра — это триггер специального типа со входами S и R , переключаемыми фронтом импульса. Вход S возбуждается старшим разрядом счетчика по модулю 9, когда в регистре сформировалось новое слово. Вход R возбуждается при передаче содержимого сдвигового регистра в буфер. Триггер заполненности буфера — это обычный D -триггер. Он сбрасывается внешним стробирующим сигналом, когда содержимое буфера считывается извне, и устанавливается в 1 (синхримпульсом при $D = 1$), когда в буфер загружается содержимое сдвигового регистра.

Наличие буферов в источнике и получателе позволяет каналу работать с максимальной скоростью, предоставляя внешним устройствам время (равное времени передачи 10 бит) для того, чтобы выдать или принять очередное слово. Когда в передаче участвует компьютер, важно иметь это дополнительное время, поскольку, для того чтобы откликнуться на информацию о состоянии, например «буфер источника пуст» или «буфер

получателя заполнен», компьютеру нужно время для выполнения соответствующих программ.

Отметим, что источник и получатель синхронизируется от физически разных генераторов. Они должны быть настроены на как можно более близкие частоты. Если эти частоты будут отличаться, то при передаче слова будет наблюдаться постепенный уход моментов срабатывания схем в получателе от середин интервалов, в которых передаются отдельные биты. От переднего фронта стартового бита схемы в получателе не должны уходить более чем на половину интервала ко времени приема 10-го бита. Иначе они вообще уйдут за пределы интервала 10-го бита. Это означает, что различие в частотах не должно превышать 5 %.

Промышленность выпускает модули, выполненные в виде интегральных схем и содержащие управляющие схемы, подобные описанным для работы с последовательными синхронными и асинхронными каналами как на стороне источника, так и на стороне получателя. Для организации двунаправленного канала на каждом его конце нужно иметь пару модулей двух разных типов.

Передающий модуль, называемый **трансмисмиттером**, на каждой стороне соединяется с принимающим модулем, называемым **ресивером** на противоположной стороне. Для тех случаев, когда каналом является носитель информации (например, кассетная магнитная лента), выпускаются модули, включающие схемы обоих типов.

Существует много разновидностей подобных модулей, и часто они содержат компоненты, повышающие их универсальность. Один из модулей для асинхронной передачи, называемый **универсальным асинхронным ресивером/трансмисмиттером (UART)**, можно запрограммировать для работы с 5-, 6-, 7- и 8-битовыми словами, с контролем по четности, по нечетности или без контроля, с одним или двумя стоповыми битами.

Программирование осуществляется подачей на определенные управляющие линии потенциала высокого или низкого уровня. В модуле предусмотрены дополнительные линии состояния, на которые выдаются сигналы о возможных ошибках, например «неверная четность», «ошибка кадра» (отсутствие стопового бита в тот момент, когда он должен быть), «перегрузка буфера» (данные поступают в трансмисмиттер слишком быстро или забираются из ресивера слишком медленно). Аналогичный модуль существует для синхронного режима передачи под названием **универсальный синхронный ресивер/трансмисмиттер (USRT)**.

Подобные модули выпускаются и для работы с конкретными микропроцессорами. Они обычно имеют тристабильные выход-

ные линии, и поэтому их удобно подключать в качестве портов ввода-вывода. Выбор той или иной программируемой возможности в некоторых модулях осуществляется путем записи управляющих слов в специальные внутренние регистры, а не внешним монтажом управляющих линий. За счет этого удастся сэкономить число внешних контактов модуля.

Для подключения к каналу, соединяющему источник с получателем, часто используются интерфейсные схемы, преобразующие электрические сигналы в нужную форму. Для двунаправленных асинхронных каналов между терминалами и компьютером наиболее часто применяются интерфейсные схемы двух стандартных типов. В одной, предназначенной для телетайпов и подобных им устройств, логическая 1 кодируется током около 20 мА, а логический 0 — отсутствием тока. В другой, входящей в стандарт EIA RS232C, логическая 1 кодируется напряжением $-5В$, а логический 0 — напряжением $+5В$.

Для асинхронных каналов с большой протяженностью типа телефонных линий или внешних запоминающих устройств типа магнитных лент применяются модуляторы и демодуляторы различных видов. В частности, существуют модули, называемые **модемами**, содержащие модулятор и демодулятор для частного кодирования данных при работе с телефонными каналами. В таких модемах для передачи в каждом направлении используется своя полоса частот, поэтому одного телефонного канала достаточно для работы в обоих направлениях.

В одной полосе единица обычно кодируется частотой 1270 Гц, а ноль — 1070 Гц. В другой полосе единицы соответствует частота 2225 Гц, а ноль — 2025 Гц. Модемы нужны на обоих концах двунаправленного канала. С помощью тех же модемов можно асинхронно записывать данные на обычный магнитофон для записи звука, поскольку указанные выше частоты лежат в звуковой полосе.

27.9. СЕКЦИОНИРОВАННЫЕ МИКРОПРОЦЕССОРЫ

Как мы уже говорили, микрокомпьютер — это модульная система, в которую входят микропроцессор, а также память и устройства ввода-вывода. Микропроцессор для достижения большей гибкости в свою очередь может быть разбит на подмодули. В некоторых случаях часть микропроцессора, обрабатывающая данные, отделяется от части, обрабатывающей команды и управляющей последовательностью их выборки. Часть, обрабатывающая данные и состоящая из арифметико-логического устройства и различных регистров данных, в этом случае сама разбивается на несколько идентичных модулей. Каждый такой модуль, называемый **микропроцессорной секцией**, состоит из

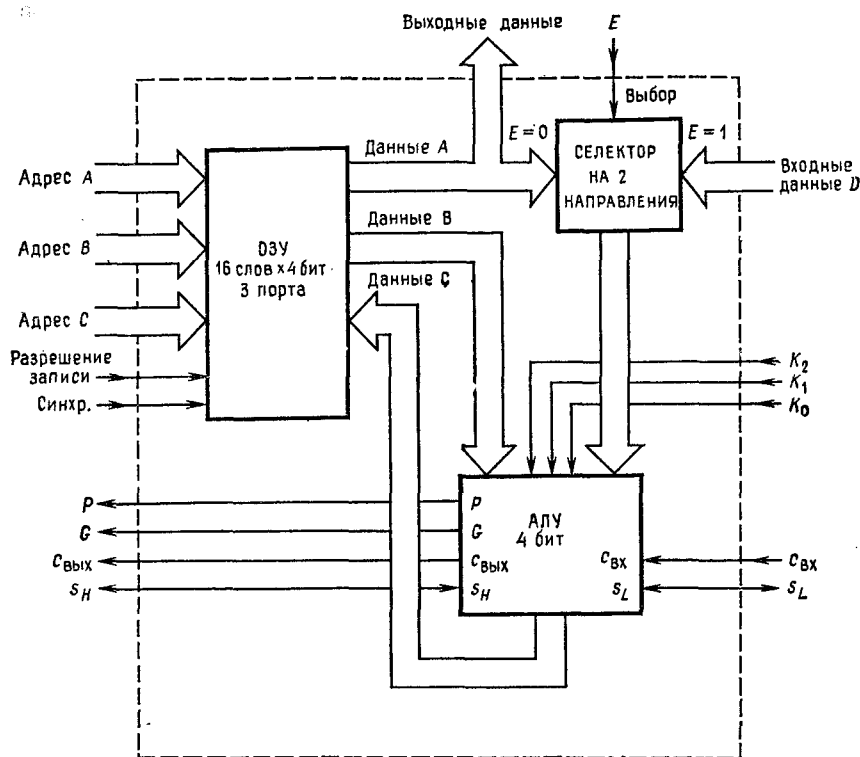


Рис. 27.28. Модель микропроцессорной секции.

арифметико-логического устройства и различных регистров определенной разрядности.

Промышленность выпускает большое количество разнообразных микропроцессорных секций, отличающихся структурой регистров и набором арифметических и логических функций. Поэтому практически невозможно построить модель, которая бы включала все многообразие существующих частных решений. Тем не менее, чтобы проиллюстрировать общие концепции, мы рассмотрим упрощенную, но достаточно представительную модель микропроцессорной секции.

Модель, показанная на рис. 27.28, включает четырехбитовое АЛУ и «сверхоперативную», или рабочую, память на 16 слов по 4 бит. АЛУ выполняет восемь различных арифметических и логических действий над двумя четырехбитовыми операндами А и В и дает четырехбитовый результат. Выполняемое действие определяется тремя управляющими линиями K₀, K₁ и K₂ в соответствии с табл. 27.3. Шесть линий, связанных с

АЛУ, позволяют соединить секции в каскады. Четыре линии $c_{\text{вых}}$, $c_{\text{вх}}$, P и G служат для распространения переносов при операциях сложения и вычитания; две оставшиеся линии s_H и s_L используются при передаче информации из секции в секцию при операциях сдвига.

Таблица 27.3. Операции, выполняемые АЛУ микропроцессорной секции

K_2	K_1	K_0	Операция	Эффект операции
0	0	0	Сложение A и B	Результат $= A + B + c_{\text{вх}}$, $c_{\text{вых}} =$ $=$ конеч. перенос
0	0	1	Вычитание B из A	Результат $= A - B - c_{\text{вх}}$, $c_{\text{вых}} =$ $=$ конеч. заем
0	1	0	И (A и B)	$r_i = a_i \wedge b_i$
0	1	1	ИЛИ (A и B)	$r_i = a_i \vee b_i$
1	0	0	ИСКЛ. ИЛИ (A и B)	$r_i = a_i \oplus b_i$
1	0	1	Сдвиг A влево	$r_i = a_{i-1}$ $i \neq 0$, $r_0 = s_L$, $s_H = a_3$
1	1	0	Сдвиг A вправо	$r_i = a_{i+1}$ $i \neq 3$, $r_3 = s_H$, $s_L = a_0$
1	1	1	Передача A	Результат $= A$

Обозначения: $A = a_3a_2a_1a_0$, $B = b_3b_2b_1b_0$; результат $= r_3r_2r_1r_0$.

В частности, $c_{\text{вх}}$ — это входная линия, несущая значение переноса или заема в младший бит, тогда как $c_{\text{вых}}$ — это выходная линия, несущая значение переноса из старшего бита или значение заема в старший бит. Две выходные линии P и G дают добавочную информацию о переносах, позволяющую соединять в каскады микропроцессорные секции вместе со схемами ускоренного переноса. Мы будем предполагать, что при реализации двух арифметических операций применяется методика ускоренного переноса.

Линии s_H и s_L двунаправленные. Линия s_H содержит значение «вдвигаемого» слева бита при сдвиге вправо и «выдвигаемого» налево бита при сдвиге влево. Линия s_L содержит значение «вдвигаемого» справа бита при сдвиге влево и значение «выдвигаемого» направо бита при сдвиге вправо.

Предполагается, что рабочая память на 16 слов по 4 бит имеет три порта. Каждый порт представляет собой совокупность из четырех линий для данных и четырех адресных линий, причем через каждый порт можно обратиться к любому из 16 слов независимо от обращений через другие порты. Через порты A и B данные читаются из рабочей памяти, а через порт C — записываются в нее.

На выходах каждого из двух портов чтения непрерывно отслеживаются значения тех слов, адреса которых поданы на соответствующие адресные линии. Содержимое памяти изменя-

ется только через порт C . Управляют записью данных через порт C две линии: «разрешение записи» и «синхр». Если линия «разрешение записи» содержит логическую 1, то запись слова по адресу, заданному на адресных линиях, происходит по заднему фронту синхронимпульса.

Два порта чтения A и B обеспечивают АЛУ двумя операндами A и B соответственно, а порт C получает результат операции из АЛУ. Операнд A в АЛУ может также поступать извне. Выбор порта A или внешних линий D в качестве источника операнда A осуществляется селектором на два направления. Управляет этим селектором внешняя линия E : внешние данные выбираются при единичном сигнале в этой линии. И наконец, 4 выходные линии данных подсоединены непосредственно к порту A рабочей памяти.

В итоге мы видим, что микропроцессорная секция способна выполнять операции над данными, находящимися в сверхоперативном ЗУ, и туда же возвращать результаты. Кроме того, операнд может поступать извне по четырем входным линиям, и данные из рабочей памяти могут быть переданы во вне по четырем выходным линиям. Выполняемая операция, источники операндов и место, куда помещается результат, определяются входящими в секцию управляющими линиями. Подавая соответствующие сигналы на эти линии, внешние схемы могут управлять операциями в микропроцессорной секции. Таким образом, одну и ту же микропроцессорную секцию можно использовать в микрокомпьютерах с различной структурой и функциями. В этом заключается существенно большая гибкость и универсальность микропроцессорных секций по сравнению с законченными одномодульными микропроцессорами.

Еще большую гибкость дает возможность объединять секции в каскады, чтобы обеспечить требуемую точность операций над данными. Несколько микропроцессорных секций можно объединить, связав линии сдвигов и переносов. Линия сдвига s_H одной секции всегда соединяется с линией s_L соседней левой секции. Что касается переносов, то здесь возможны два варианта. Простейший состоит в соединении линии $c_{вых}$ с линией $c_{вх}$ соседней левой секции. Это приведет к покаскадному последовательному распространению переносов от секции к секции.

Для того чтобы реализовать в полном объеме преимущества, заложенные в описанной микропроцессорной секции, следует воспользоваться схемой генератора сквозных переносов. В этом случае схема генератора переносов получает сигналы по линиям P и G от каждой секции и выдает сигналы на линии $c_{вх}$ в каждую секцию. Линии $c_{вых}$ не используются. Вариант каскадирования секций с использованием генератора сквозных переносов приведен на рис. 27.29,

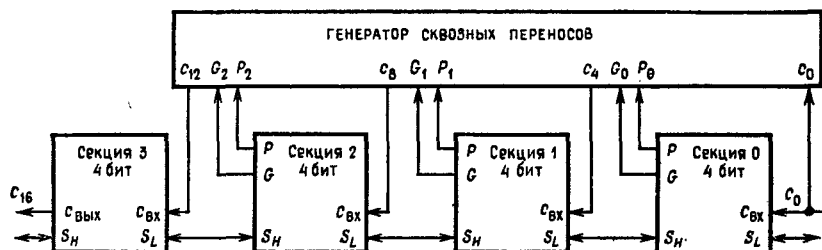


Рис. 27.29. Каскадирование микропроцессорных секций с использованием генератора сквозных переносов.

Чтобы сформировать полный микропроцессор из секций, нужны дополнительные компоненты для управления потоком команд и для их дешифрации. В число дополнительных компонентов должны входить программный счетчик, регистр команды, блок управления и синхронизации, дешифратор команды, возможно, стек и другие регистры для обеспечения интерфейсов с памятью и устройствами ввода-вывода.

Вообще говоря, дополнительные компоненты должны обеспечивать чтение команды из главной памяти в соответствии с программным счетчиком и занесение ее в регистр команды. Дешифратор команды вместе с блоком управления и синхронизации по содержимому регистра команды должен сформировать сигналы на управляющих и других входных линиях микропроцессорных секций, чем обеспечивается выполнение команды. Программный счетчик должен увеличиваться после чтения каждой команды и соответствующим образом изменяться при переделах управления.

Реализация иллюстративного микропроцессора на микропроцессорных секциях

В качестве примера рассмотрим, как можно было бы реализовать наш иллюстративный микропроцессор с применением модельных микропроцессорных секций. Для получения восьмибитового слова достаточно соединить две секции. При этом мы имеем АЛУ и 16 общих регистров. Остальные компоненты должны содержать 16-битовый программный счетчик, стек из 64 слов по 16 бит, 8-битовый регистр команды, 16-битовый регистр адреса данных, дешифратор команды, триггер переноса, необходимые схемы синхронизации и управления, а также различные шинные вентили. Основные связи между этими компонентами показаны на рис. 27.30.

На рисунке блок с названием «общие логические схемы» управляет потоками данных между различными компонентами.

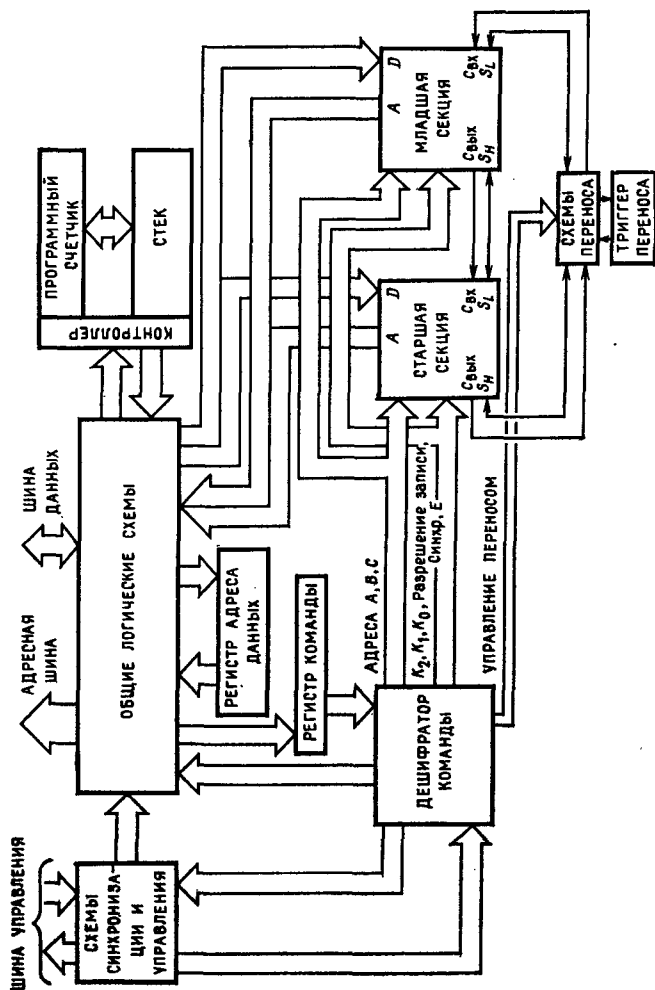


Рис. 27.30. Реализация иллюстративного микропроцессора на микропроцессорных секциях.

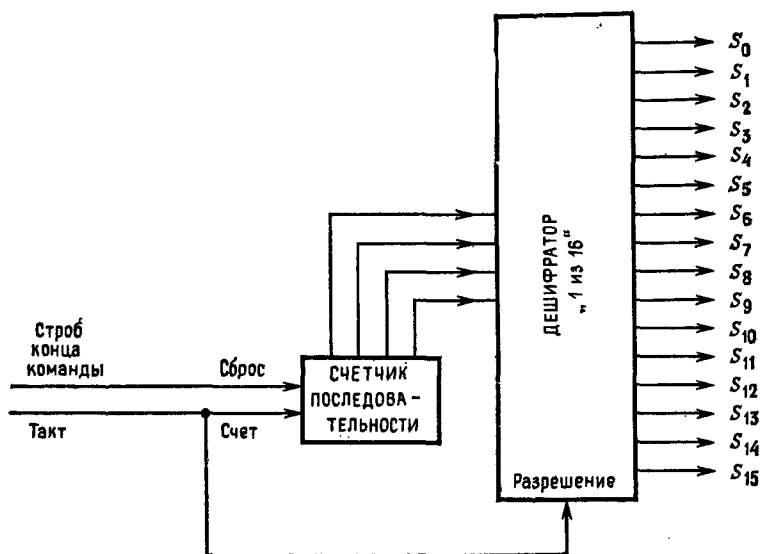


Рис. 27.31. Счетчик и дешифратор временной последовательности.

Временные сигналы он получает от блока, названного «схемы синхронизации и управления».

Блок «дешифратор команды» по содержимому регистра команды формирует управляющие сигналы в блок общих логических схем и в микропроцессорные секции. В микропроцессорные секции от дешифратора команды поступают адреса A , B и C операндов и результата в рабочей памяти, а также шесть управляющих сигналов: K_2 , K_1 , K_0 , «разрешение записи», «синхр» и E . В блок общих логических схем от дешифратора команды поступает либо информация, касающаяся изменения программного счетчика (при командах переходов), либо информация о передачах данных в секции или из них. Моменты поступления той или иной информации зависят от сигналов, приходящих от схем синхронизации и управления.

Как было только что сказано, схемы управления и синхронизации задают моменты срабатывания других блоков. Эти же схемы получают входные сигналы и формируют выходные сигналы управляющей шины. Центральным компонентом в блоке схем синхронизации и управления является четырехбитовый счетчик, задающий временную последовательность. Этот счетчик подключен к входной управляющей линии «такт» и увеличивается при каждом тактирующем импульсе. В конце каждой команды счетчик сбрасывается в нуль. По состоянию счетчика определяются моменты подачи синхроимпульсов на каждом

шаге выполнения команды. Для этого четыре бита счетчика поступают на вход дешифратора «1 из 16», как показано на рис. 27.31.

Сигналы на выходных линиях этого дешифратора, обозначенных S_1 , стробируются тактовыми импульсами. Таким образом, в каждом такте выполнения команды импульс появляется только на одной выходной линии дешифратора, соответствующей номеру тактирующего импульса от начала командного цикла. Сигналы S_1 поступают в блок общих логических схем и в блок дешифратора команды. В двух этих блоках по сигналам S_1 в комбинациях с другими логическими сигналами формируются сигналы, стробирующие различные передачи между регистрами и триггерами. Эти же сигналы S_1 в самой блоке схем синхронизации и управления используются для формирования сигналов в линиях шины и управления.

Для того чтобы проиллюстрировать работу схемы в целом, рассмотрим выполнение двух команд. Первые два такта одинаковы для всех команд, поскольку в это время код операции команды еще только выбирается из главной памяти. Импульс S_0 (передний его фронт) используется для установки адреса команды на адресные шины. Затем устанавливается логическая 1 на линии «чтение». В результате первый байт команды подается из главной памяти на шину данных. Затем по заднему фронту S_1 стробируется прием содержимого шины данных на регистр команды. Дешифратор команды после этого определяет тип команды и сообщает блоку общих логических схем о необходимости выбрать дополнительные байты, если команда состоит более чем из одного байта.

Предположим, что выполняется команда ADC (сложение с регистром и переносом). Поскольку это однобайтовая команда, дополнительные байты не выбираются. При следующем тактирующем импульсе микропроцессорные секции осуществляют выполнение команды. Оно сводится к сложению содержимого общего регистра R с содержимым аккумулятора, т. е. регистра 0. Результат должен быть направлен в аккумулятор.

Чтобы осуществить все это, дешифратор команд подает значение 0000 в качестве адресов A и C и значение R в качестве адреса B . На управляющие линии K_2 , K_1 и K_0 подается комбинация 000, означающая сложение, а на линию E — логический 0, означающий, что второй операнд нужно взять из порта A . Блок схем переноса получает указание подать значение триггера переноса на линию $c_{вх}$. И наконец, на линии «разрешение записи» устанавливается логическая 1.

После всех этих установок импульс S_2 подается на линию «синхр» микропроцессорных секций. Это приводит к записи вычисленной суммы в аккумулятор (т. е. в общий регистр 0).

В тот же момент импульс подается в блок схем переноса для стробирования записи конечного переноса на триггер переноса. Одновременно происходит сброс счетчика временной последовательности, поскольку выполнение команды завершено.

В качестве второго примера рассмотрим команду безусловного перехода, в выполнении которой микропроцессорные секции никак не участвуют. После поступления первого байта на регистр команды дешифратор сообщает общим логическим схемам о необходимости считать из главной памяти еще два байта. В результате второй байт выбирается из памяти и запоминается в старшей половине регистра адреса данных по импульсам S_2 и S_4 (они соответствуют первому и второму тактовым импульсам второго машинного цикла). Третий байт выбирается по импульсам S_6 и S_7 (соответствующим первому и второму импульсам третьего машинного цикла) и запоминается в младшей половине регистра адреса данных. Выполнение команды завершается передачей нового содержимого регистра адреса данных на программный счетчик по импульсу S_8 . Этим же импульсом S_8 сбрасывается счетчик временной последовательности.

О применении микропроцессорных секций

Можно сказать, что, вообще говоря, микропроцессорные секции предоставляют более гибкие возможности по сравнению с одномодульными микропроцессорами. В частности, секционирование позволяет легко получить требуемую точность представления данных. Более того, секции можно сгруппировать так, чтобы одновременно обрабатывались разные компоненты векторной величины, и тем самым создать процессор, работающий с векторами, а не со скалярами. Меняя состав и функции других логических блоков, можно варьировать систему команд микропроцессора.

Микропроцессоры, использующие секции, обычно оказываются и более быстродействующими по сравнению с одномодульными благодаря меньшей степени интеграции входящих в них схем. Тем не менее необходимость подключать дополнительные схемы нужно отнести к недостаткам секционирования. Чтобы как-то компенсировать этот недостаток, разработаны и выпускаются модули, управляющие последовательностью команд. Чтобы создать законченный микропроцессор из микропроцессорных секций, к такому модулю нужно добавить только регистр команды, дешифратор и другие регистры, специфичные для области приложения. Оставляя регистр команды и дешифратор вне модуля, мы сохраняем преимущества гибкости и универсальности.

Общепринятый подход при создании модулей, управляющих последовательностью команд, сводится к выполнению команды как последовательности микрокоманд. Микрокоманды размещаются в специальной памяти, и их выборка и выполнение осуществляются аналогично обычным командам. Очередность выборки микрокоманд определяется модулем, управляющим последовательностью команд. Части каждой микрокоманды декодируются дополнительными схемами и задают сигналы, поступающие в микропроцессорные секции. Те же микрокоманды используются для работы с внешним программным счетчиком и регистром команды в процессе обработки обычных команд.

Обычные команды (те, которые входят в набор команд микропроцессора) размещаются в главной памяти и выбираются из нее на регистр команды в соответствии с программным счетчиком. Осуществляет это последовательность из нескольких микрокоманд. После этого содержимое регистра команды используется как адрес микрокоманды, которая специально написана для выполнения данной обычной команды. Таким образом, мы имеем дело с компьютером внутри компьютера, причем внутренний компьютер программируется при помощи микрокоманд таким образом, чтобы выполнять команды внешнего компьютера.

Следует подчеркнуть, что **микропрограммирование** означает не программирование для микрокомпьютера, а реализацию набора команд некоторого компьютера в виде совокупности микропрограмм.

27.10. СИНХРОНИЗАЦИЯ МИКРОПРОЦЕССОРОВ

В отличие от нашего иллюстративного микропроцессора многие промышленные микропроцессоры нуждаются более чем в однофазной синхронизации. Импульсы в каждой отдельной синхронизирующей линии (в каждой фазе) имеют одну и ту же частоту. Поэтому взаимные временные сдвиги импульсов в разных фазах постоянны. Обычно разные типы передач в микрокомпьютере синхронизируются разными фазами. Например, регистры процессора могут быть построены на двухтактных MS-триггерах. В этом случае одна фаза стробирует прием данных в ведущие секции триггеров, а другая — передачу из ведущих в ведомые.

Одна из возможных схем формирования двух неперекрывающихся фаз синхроимпульсов с использованием одного генератора импульсов показана на рис. 27.32. Генератор подключен к входу синхронизации триггера T -типа, на входе T которого постоянно поддерживается логическая 1. Таким образом триггер переключается при каждом импульсе от генератора. Как прямой,

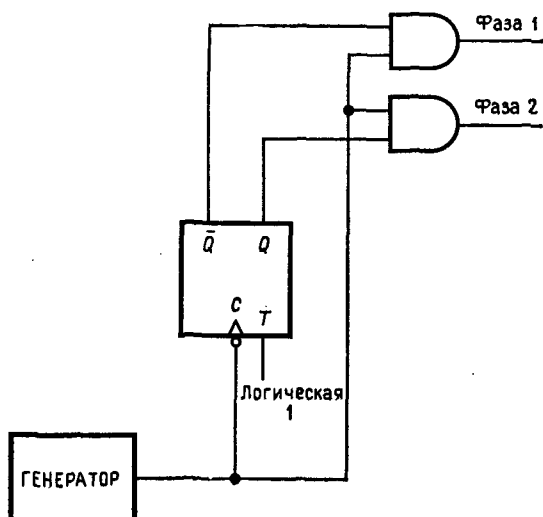


Рис. 27.32. Схема делителя фаз синхроимпульсов.

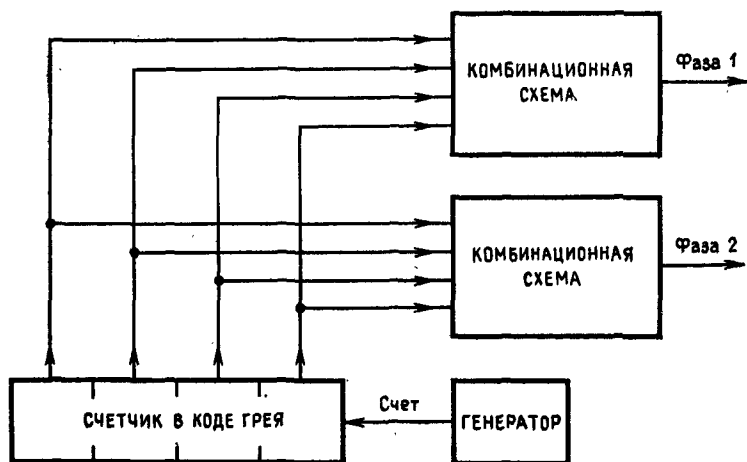


Рис. 27.33. Обобщенная схема делителя фаз синхроимпульсов.

так и инверсный выходы триггера поданы на вентили И, на вторые входы которых поступают импульсы от генератора. Выходы вентилях и являются линиями фаз. Когда выход Q равен логическому 0, импульс от генератора вызывает импульс на линии фазы 1. Когда выход Q равен логической 1, импульс от генератора вызывает импульс на линии фазы 2,

Таким образом, импульсы появляются на линиях фаз поочередно. Промежутки между импульсами разных фаз определяются промежутком между импульсами генератора и, следовательно, может настраиваться в соответствии со специфическими требованиями конкретного микропроцессора. Частота импульсов в каждой фазе равна половине частоты генератора. Длительность импульсов фаз совпадает с длительностью импульсов генератора.

Более сложные временные соотношения между импульсами двух фаз можно получить с помощью схемы на рис. 27.33. В этой схеме импульсы от генератора поданы на счетный вход счетчика в коде Грея. Синхроимпульсы в каждой фазе могут иметь произвольное относительное расположение и длительность за счет комбинирования последовательных состояний счетчика. При этом используются специальные «помехоустойчивые» (hazard-free) комбинационные схемы¹⁾.

Счетчик в коде Грея, применяющийся вместо обычного двоичного счетчика, позволяет уменьшить помехи, возникающие при одновременном переключении нескольких разрядов. При четырехбитовом счетчике частота генератора должна быть в 16 раз выше частоты импульсов синхронизации в обеих фазах. Приведенная схема допускает простое обобщение на случай большего числа фаз путем увеличения числа комбинационных схем.

¹⁾ С устройством таких схем читатель может познакомиться по книгам, посвященным теории переключательных схем. — *Прим. ред.*

ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ МИКРОКОМПЬЮТЕРОВ

С. Мясковски

28.1. ВВЕДЕНИЕ

Операционная система, несмотря на то что она представляет собой лишь особый тип программного обеспечения, является одной из наиболее важных компонент вычислительной системы. Все операционные системы независимо от того, для какого класса ЭВМ они предназначены: для универсальных компьютеров, мини-компьютеров или микропроцессорных систем, осуществляют в сущности интерфейсные функции между логическим уровнем пользователя и физическим уровнем аппаратуры.

Кроме того, программы операционной системы обеспечивают управление ресурсами вычислительного комплекса, а именно процессором, памятью и устройствами ввода-вывода.

28.2. КОМПОНЕНТЫ ОПЕРАЦИОННЫХ СИСТЕМ

Современные операционные системы для микрокомпьютеров часто проектируются и создаются по модульному принципу, что обеспечивает их гибкость, а также мобильность, т. е. способность к перемещению с одних компьютеров на другие. Можно выделить следующие типичные уровни таких модульных систем:

Ядро. Наиболее близкий к аппаратуре уровень. Модули этого уровня выполняют базовые функции операционной системы, такие, как, например, планирование.

Управление ресурсами. На этом уровне осуществляется распределение ресурсов согласно запросам других уровней. Здесь происходит координирование обращений к памяти для устройств ввода-вывода и микропроцессора.

Физический уровень ввода-вывода. Этот уровень часто реализуется в виде набора драйверов устройств и обеспечивает управление потоками данных между памятью компьютера и внешними устройствами, такими, как, например, консоль, диски, устройства печати, а также устройства параллельного или последовательного обмена.

Логический уровень ввода-вывода. Благодаря наличию этого уровня возможно формировать запросы от операционной системы к устройствам обмена, не используя физических характеристик этих устройств. С этого уровня данные передаются на физический уровень и наоборот.

Система управления файлами. Модули этого уровня обслуживают справочник информационных файлов, находящихся на периферийных устройствах, и обеспечивают управление потоками данных между физическим и логическим уровнями ввода-вывода.

Человеко-машинный интерфейс. На этом уровне реализуется связь пользователя с системой. Он является самым внешним и представляет собой как бы «раковину», заключающую в себе всю систему. На этом уровне производится интерпретация символьной формы команд, вводимых оператором с консоли, а также перевод в символьный вид сообщений и данных, поступающих на дисплей.

Прикладное программное обеспечение. Этот уровень обеспечивает взаимодействие между операционной системой, языками высокого уровня, вспомогательными программами (утилитами) и программами пользователей.

28.3. ОСОБЕННОСТИ ОПЕРАЦИОННЫХ СИСТЕМ ДЛЯ МИКРОКОМПЬЮТЕРОВ

До недавнего времени таким особенностям операционных систем, как мобильность и совместимость с программами, созданными для других компьютеров, уделялось мало внимания. В универсальных комплексах выполнялась настройка операционных систем на конкретные и определяемые пользователем условия эксплуатации, зависящие, в частности, от особенностей прикладных задач.

С появлением общедоступных микропроцессорных систем (а именно, персональных компьютеров) важность проблем мобильности и совместимости значительно возросла. Пользователи микропроцессорной системы вправе считать, что программное обеспечение, получаемое ими от поставщиков (во многих случаях обеспечивающих лишь минимальный уровень их требований), будет нормально работать на приобретенных системах. Способ, который позволяет удовлетворить условиям массовой продажи микропроцессорных систем, заключается в создании стандартного интерфейса между прикладным программным обеспечением и аппаратурой. Как будет показано далее, в производстве микрокомпьютеров предпринимаются определенные шаги в этом направлении.

28.4. МИКРОКОМПЬЮТЕРЫ ПРОТИВ УНИВЕРСАЛЬНЫХ ЭВМ

Операционные системы для микрокомпьютеров и для универсальных ЭВМ в своих основных функциях имеют очень небольшие отличия. Оба типа операционных систем обеспечивают необходимый человеко-машинный интерфейс и управление ресурсами. Главное же отличие между ними заключается в том, что для микропроцессорных систем требуется управлять малым количеством ресурсов, т. е. меньшим объемом памяти, сравнительно небольшим набором устройств ввода-вывода и более медленным микропроцессором, имеющим ограниченный набор команд. Кроме того, такие возможности операционных систем для универсальных ЭВМ, как пакетная обработка и одновременная работа большого числа пользователей, для малых компьютерных систем не нужны.

Модули операционной системы, предназначенные для управления ограниченными ресурсами, занимают небольшой объем памяти, например в восьмиразрядных микропроцессорах можно непосредственно обращаться лишь к ограниченному адресному пространству, редко превышающему 64К. Именно поэтому операционные системы общего назначения, занимающие сотни килобайт адресного пространства, невозможно использовать в малых системах. Следовательно, несмотря на ограниченные требования к управлению ресурсами микропроцессорной системы, модули операционной системы должны быть максимально оптимизированы, чтобы они могли быть размещены в минимальном объеме памяти.

Следующая особенность проектирования операционных систем для микрокомпьютеров заключается в том, чтобы обеспечить возможность работы «неподготовленных» пользователей, какими являются большинство пользователей персональных компьютеров, не обладающих достаточным опытом в программировании. Из этого требования вытекает необходимость реализации таких дополнительных возможностей, как выдача простых «подсказок», вспомогательных сообщений, а также средств автоматического обнаружения и исправления ошибок.

Стоимость микрокомпьютеров на 16-разрядных микропроцессорах стала достаточно низкой, поэтому такие компьютеры стали доступны широкому кругу пользователей малых вычислительных систем. Использование 16-разрядных микропроцессоров сняло ограничения, возникающие из-за малых размеров адресного пространства. Выиграв в памяти и повысив вычислительную мощность 16-разрядных микропроцессоров, поставщики, выпускающие микрокомпьютеры, разрабатывают для них операционные системы, соперничающие по возможностям и сложности с операционными системами универсальных и мини-

компьютеров. Действительно, как это происходит и с аппаратурой, демаркационные линии, разделяющие операционные системы этих трех классов компьютеров, становятся все менее явными. Разработаны операционные системы для микрокомпьютеров, имеющие такие возможности и сложность, какие были до недавнего времени свойственны лишь операционным системам более мощных типов компьютеров. Эти системы обеспечивают многопользовательский и мультизадачный режим работы; защиту данных от сбоев и использования их другими пользователями или задачами; разнообразные средства для повышения эффективности программирования; индексно-последовательный метод доступа к данным на томах прямого доступа (ISAM); автоматические средства обнаружения и исправления ошибок, более эффективные методы хранения данных, а также средства для работы неподготовленных пользователей. Кроме того, разработаны и более сложные операционные системы для 8-разрядных микрокомпьютеров, которые благодаря страничной организации позволяют работать с памятью больших размеров, чем принято для компьютеров этого класса, и обеспечивают многие из перечисленных выше возможностей.

28.5. ПРОБЛЕМЫ СОВМЕСТИМОСТИ И МОБИЛЬНОСТИ ОПЕРАЦИОННЫХ СИСТЕМ

Операционные системы для малых компьютеров обычно называют дисковыми операционными системами (DOS), поскольку стандартными устройствами внешней памяти являются сменные гибкие или постоянные винчестерские диски. Хотя фирмы, выпускающие персональные компьютеры на микропроцессорах, за последние годы прошли большой путь, до сих пор проблема совместимости и мобильности остается во многом нерешенной. Наиболее крупные фирмы-изготовители персональных компьютеров, вообще говоря, имеют собственные операционные системы, разработанные специально для выпускаемых ими компьютеров. Наглядными примерами таких систем являются Apple DOS для персонального компьютера Apple II, SOS для Apple III, а также несколько версий TRSDOS для серии персональных компьютеров фирмы Radio Shack. Перечисленные системы созданы с тем расчетом, чтобы оптимально учесть возможности и характерные свойства каждого из перечисленных персональных компьютеров. Но в них нет единого стандартного интерфейса с пользователем, а также стандартного языка команд. Следовательно, прикладные программы, разработанные на одной системе, не могут быть переиснены на другую систему без дополнительной переработки.

Поскольку для большинства прикладных программ требуется использование копии операционной системы на дистрибутивном носителе, возникает другая проблема, связанная с тем, что некоторые фирмы — изготовители компьютеров (например, Radio Shack) требуют дополнительной выплаты по лицензии для любых прикладных программ, содержащих их операционные системы. Некоторые разработчики обходят эту проблему, создавая независимые операционные системы, разрабатываемые специально для того, чтобы совместить программы с аппаратурой, обслуживаемой фирменной операционной системой. Использование таких систем несложно и доступно прикладным программистам. Сложность задачи во многом зависит от сложности аппаратуры и доступности информации о системе.

Много систем такого рода создано для операционной системы TRS-80, за каждую копию дистрибутивного носителя которой фирмы Radio Shack требует дополнительной выплаты. Примерами систем совмещения с TRS-80 могут служить LDOS, NEWDOS и DOSPLVS.

Совершенно другой подход к распространению своих операционных систем практикует фирма Apple Computer, выпускающая самые популярные персональные компьютеры. Хотя фирма утверждает, что на каждой модели компьютера элементы операционной системы (DOS) реализованы по-своему, буквально тысячи пакетов прикладных программ, разработанных для компьютера Apple II, внедряются на Apple III. Пользователь может устанавливать операционную систему этой фирмы вместе со своим прикладным программным обеспечением без дополнительной платы за лицензию. Фирма от этого только выигрывает, так как увеличение общего объема программного обеспечения для ее компьютеров приводит к увеличению спроса на них.

28.6. ОПЕРАЦИОННАЯ СИСТЕМА CP/M

Существует операционная система, фактически ставшая стандартной для микрокомпьютеров. Это система CP/M, разработанная в 1975 г. фирмой Digital Research, Inc. Система CP/M используется в вычислительных системах на базе микропроцессоров Z80, 8080 и 8085, а после переделки ее на 16-разрядный вариант может применяться и для микропроцессоров 8086.

Поскольку система CP/M является первой универсальной операционной системой, созданной для микропроцессоров, ее создание ликвидировало наконец этот явный пробел и привело к широкому ее распространению. В течение нескольких лет система CP/M была единственной универсальной системой и быстро приобрела статус стандарта. В то же время она под-

верглась ряду модификаций и усовершенствований. Несмотря на большую популярность, эта система обладает рядом недостатков.

Тем не менее, поскольку на многих микропроцессорах используется система CP/M, охарактеризуем ее более подробно. Дальнейшее изложение можно рассматривать как общее введение в микрокомпьютерные операционные системы, поскольку концепции CP/M во многом свойственны и другим операционным системам.

Введение в CP/M

Система CP/M представляет собой микрокомпьютерную операционную систему, предназначенную для обслуживания одного пользователя в однозадачном режиме. Как правило, система CP/M располагается в старших адресах памяти и делится на три основных модуля: BIOS (basic input/output system) — базовую систему ввода-вывода, BDOS (basic disk operating system) — базовую дисковую операционную систему и CCP (console command processor) — процессор команд, поступающих с консоли. Память, используемая задачей пользователя, называемая TPA (transient program area), размещается с младших адресов доступного адресного пространства. Рассмотрим каждую из частей подробнее.

Модуль BIOS представляет собой аппаратно-зависимый модуль, выполняющий функции интерфейса между аппаратурой микрокомпьютерной системы и BDOS. Он является единственным модулем, зависящим от спецификаций аппаратуры, и содержит подсистемы для ввода-вывода символьной информации и для управления дисками. Этот модуль обеспечивает такие функции, как определение адреса для прямого доступа к памяти, размещение дорожек и секторов и управление чтением-записью данных. Настраиваемые версии CP/M выпускаются со стандартным модулем (предназначенным для системы автоматизированной разработки микрокомпьютеров MDS-80 фирмы Intel), который можно приспособить с помощью определенных операций для использования в других системах. Во многих случаях BIOS настраивается на определенную среду с помощью систем автоматизированной разработки микрокомпьютеров.

Модуль BDOS является основным модулем системы CP/M. Он отвечает за управление всей памятью, процессором, а также за выполнение операций ввода-вывода. Этот модуль работает при поддержке модуля BIOS.

Модуль загрузки располагается в младших адресах памяти, отводимой задаче пользователя, и обеспечивает ввод и инициализацию программы, необходимой для передачи управления консолью модулю CCP.

Функцию интерфейса системы с «реальным миром» пользователя выполняет модуль ССР. Когда система находится на уровне команд, модуль ССР ждет команд, набираемых на клавиатуре пользователем. Этот модуль обеспечивает поиск файлов на диске, согласно командам пользователя, выполнение этих команд, а также запись информации на диск.

Пользователь осуществляет взаимодействие с модулем ССР вводом стандартных команд, оканчиваемых обычно именами файлов. Затем модуль ССР ищет файл на носителе (обычно или на гибком сменном или постоянном диске). Если файл (например, содержащий программу) найден, то он загружается в память ТРА, в противном случае возвращается сообщение об ошибке. Затем модуль ССР передает управление загруженной программе, которая начинает выполняться, используя при этом средства организации ввода-вывода, представляющие BDOS и BIOS. После завершения выполнения программы управление возвращается к модулю ССР.

Организация системного диска CP/M

Диск, на котором размещается операционная система CP/M, делится на 3 области. Первые два трека (0 и 1) используются для самой операционной системы. Поскольку эта область зашифлена, операционная система не может быть перенесена на диск обычной командой копирования. При генерации CP/M применяется специальная команда, которая обеспечивает запись операционной системы на треки 0 и 1.

Вторая область содержит оглавление диска и располагается на 2-м треке. Размер этой области динамически изменяется в зависимости от числа файлов, записанных на данном конкретном диске.

Оставшаяся часть диска является областью данных. Реальный размер этой области зависит от вида и типа дисков, используемых в системе.

Каждый файл на диске имеет FCB (file control block) — блок управления файлом, содержащий номер, имя и длину файла. Пространство на дисках заполняется блоками с минимальной длиной 1К байт для устройств с одинарной плотностью записи и 2К байт для устройств с двойной плотностью. Используемые в настоящее время версии CP/M допускают максимальный объем файла до 8 Мбайт.

Модуль CP/M содержит пять встроенных команд:

TYPE — вывод файла на дисплей

DIR — вывод оглавления диска

REN — изменение имени файла

ERA — уничтожение файла

SAVE — сохранение файла

Под управлением CP/M в область TPA могут загружаться вспомогательные программы. В их число входят:

SYSGEN — программа для записи системных файлов CP/M на диск;

PIP — программа передачи информации между периферийными устройствами;

ED — построчный редактор текстовой информации;

ASM — ассемблер;

DUMP — программа выдачи информации файла в шестнадцатеричном виде;

LOAD — программа загрузки файла;

STAT — программа формирования справочной информации о диске;

MOVCPM — программа изменения конфигурации системы;

SUBMIT — программа, выполняющая последовательность команд;

DDT — программа отладки в динамическом режиме.

Развитие системы CP/M

Несмотря на доступность и широкое распространение операционной системы CP/M, она имеет ряд серьезных недостатков, ощутимых для сегодняшних «типичных» пользователей персональных компьютеров, обладающих ограниченными знаниями о вычислительной технике. Первый недостаток заключается в том, что, поскольку эта система создавалась в начальный период развития микрокомпьютеров, она была (и продолжает оставаться) системой для опытных программистов, хорошо знакомых с ее структурой команд. Отсутствие простого, доступного для некомпетентного пользователя интерфейса представляет собой серьезную проблему по той причине, что доступность персональных компьютеров в настоящее время значительно возросла.

Чтобы использовать систему CP/M, пользователь должен изучить сложные последовательности команд этой системы, знать ее «кухню». Возможно самое существенное — это отсутствие команды типа **HELP**, выводящей на экран справочную информацию о правилах использования системы, и средств защиты от ошибок. Неопытный пользователь может совершенно беспрепятственно уничтожить информацию целого файла или

даже всего диска, не имея никаких возможностей для ее восстановления.

Отсутствие понятного для неопытного пользователя интерфейса до некоторой степени компенсировалось «надстройкой» над операционной системой, представляющей собой сравнительно недорогое программное обеспечение и обеспечивающей наглядный и простой интерфейс. Эта надстройка представляет собой расширение модуля ССР и способна воспринимать команды более естественного вида. Например, вместо команды копирования диска, имеющей вид `PIP B:=A:*.*,` можно задавать команду `COPY A TO B.` Кроме того, в расширении операционной системы обеспечены команды типа «меню», представляющие пользователю возможность выбора одного из заданного числа вариантов.

Система СР/М представляет собой (как операционная система исключительно для микрокомпьютеров) до некоторой степени компромиссную систему. Все операционные системы создаются с целью обеспечения интерфейса между пользователем и компьютером. Но если в ряде систем преследуется цель оптимального распределения ресурсов только во время выполнения команд системы, то в других системах решается задача оптимального распределения ресурсов во время выполнения программ пользователей. Поскольку система СР/М создавалась на начальных этапах развития микрокомпьютеров, она занимает промежуточное положение между указанными двумя типами систем. Законченность структуры команд СР/М, а также прямой доступ к файловой системе представляют программисту мощные средства разработки программного обеспечения. Кроме того, пять ее встроенных команд обеспечивают сравнительно наглядный (но в то же время часто трудный для изучения) сервис при работе с системой.

Мобильность операционной системы СР/М

Поскольку СР/М фактически признана стандартной операционной системой для вычислительных систем на микропроцессорах, ее приспособливают и используют во многих доступных в настоящее время персональных компьютерах. Однако система СР/М может работать только в вычислительных системах на базе микропроцессоров Intel-8080 и Z80. Поскольку в большинстве персональных компьютеров используется один из указанных микропроцессоров, серьезных проблем при применении на них системы СР/М не возникает. Однако пользователи персональных компьютеров фирмы Apple (поскольку в этих компьютерах используются микропроцессоры 6502) были лишены возможности использовать систему СР/М до тех пор,

пока не был разработан аппаратный модуль для компьютеров этого типа, обеспечивающий на них эксплуатацию этой операционной системы. Хотя подключение модуля позволило пользователям компьютеров фирмы Apple непосредственно применять на своих компьютерах сотни пакетов прикладных программ, разработанных для систем CP/M, очевидно, что тот же модуль не даст использовать программное обеспечение, созданное для микропроцессора 6502.

В заключение заметим, что CP/M является операционной системой, реализующей однозадачный режим для единственного пользователя. Когда начали использовать 16-разрядные микропроцессоры и за счет этого расширилось адресное пространство, стали появляться операционные системы для нескольких пользователей. Многопользовательская версия системы CP/M, называемая MP/M, разработана фирмой Digital Research. Но для 16-разрядных микропроцессоров версия CP/M, рассчитанная на работу многих пользователей, пока не создана.

28.7. УСОВЕРШЕНСТВОВАННЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ 8-РАЗРЯДНЫХ КОМПЬЮТЕРОВ

Широкое распространение 8-разрядных компьютеров, достигнутое в настоящее время, вызвало дальнейшее развитие и модификацию операционных систем для 8-разрядных микропроцессоров, хотя за последнее время было создано много операционных систем для 16-разрядных микропроцессоров, но они будут обсуждаться позднее. В качестве примера рассмотрим операционную систему OASIS фирмы Phase One Systems, Inc. В настоящее время создан 16-разрядный вариант этой системы. Разработчики OASIS классифицировали ее, как «коммерческую систему», поскольку она предназначена для разработчиков программного обеспечения для решения коммерческих задач, а также для пользователей, ориентированных на решение этих задач. Рассматриваемая система может быть установлена (перемещена) на микрокомпьютерах, основанных на микропроцессоре Z80, и реализует возможности, доступные до недавнего времени только на мини-компьютерах. Кроме того, система обеспечивает наглядный для неквалифицированного пользователя интерфейс, рассчитанный на решение коммерческих задач, а также дополнительные возможности, необходимые для разработки сложных прикладных программ.

Организация системы OASIS

Операционная система OASIS состоит из трех основных частей. Первая из них — NUCLEUS, аналогичная BDOS системы CP/M, реализует основные функции системы. Система

OASIS является многопользовательской системой, и поэтому ее основные функции по сравнению с системой CP/M являются более сложными и включают в себя планирование заданий, управление памятью и ресурсами для нескольких пользователей. Другая часть системы — интерпретатор командных строк CSI (command string interpreter) — аналогична CCP системы CP/M и обеспечивает выполнение функций обмена с файлами пользователя, а также управление пользователем работой системы. Прикладное программное обеспечение, представляющее третью часть системы, включает в себя более 50 системных вспомогательных программ-утилит, а также трансляторы с языков, среди которых Бейсик и макроассемблер.

Система OASIS, как и большая часть операционных систем, представляет собой в основных чертах программу управления файлами. Она обеспечивает работу с файлами, имеющими различные типы организации, в том числе и несколько видов индексной. В этой системе реализуется защита файлов и записей от несанкционированного доступа. Для этой цели применяется доступ к данным на шести уровнях привилегированности, а также использование паролей. Обеспечение защиты файлов особенно важно при использовании многопользовательского и многотерминального режимов работы, а также представляет собой эффективное средство, предохраняющее от случайного изменения или уничтожения файлов неквалифицированными пользователями.

Средства переноса операционных систем

Разработанная недавно 16-разрядная версия системы OASIS является одной из операционных систем, обеспечивающих перемещение программ для 8-разрядных микропроцессоров на 16-разрядные. Другая система такого типа — MS-DOS, представляющая собой операционную систему, обслуживающую одного пользователя в однозадачном режиме. Эта система была разработана для персонального компьютера фирмы IBM (IBM-PC) и рассчитана в основном на управление работой программ, а не на их разработку. В системе MS-DOS имеются утилиты, преобразующие объектные коды программ для микропроцессоров Z80 и 8080 в коды микропроцессора 8086, используемые в компьютере IBM-PC.

В системе MS-DOS реализован ряд перспективных для систем этого класса возможностей. Эта система занимает лишь 32К памяти ОЗУ вместо 64К, требующихся для размещения CP/M, и, кроме того, в ней реализованы средства, позволяющие адаптировать программы, написанные на языках Бейсик, Кобол, Фортран и Паскаль, для того чтобы их можно было считать на компьютере IBM-PC. В системе MS-DOS автомати-

чески отмечаются даты создания файлов, что представляет собой простое, но весьма удобное средство для пользователей и разработчиков программ, и, кроме того, в этой системе предусмотрены возможности для выявления сложных ошибок. Система MS-DOS позволяет адресоваться более чем к одному миллиарду битов на диске.

Последняя версия MS-DOS позволяет настраивать интерфейс системы и пользователя в зависимости от требований заказчика. В ней реализован фоново-оперативный режим работы, а также обеспечены средства управления вычислительными системами типа локальных сетей.

28.8. ОПЕРАЦИОННЫЕ СИСТЕМЫ ДЛЯ 16-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ

Распространение 16-разрядных вычислительных систем привело к появлению ряда операционных систем для 16-разрядных микропроцессоров. В отличие от 8-разрядных операционных систем стандартная операционная система для 16-разрядных компьютеров пока не создана и едва ли в ближайшее время появится. Основная причина этого заключается в большом разнообразии возможных применений 16-разрядных вычислительных систем, начиная от простых компьютеров, реализующих режим для одного пользователя, до сложных высокоуровневых систем, обеспечивающих многозадачный режим, обслуживающих нескольких пользователей и позволяющих быстро решать задачи коммерческого типа. Кроме того, для задач управления процессами необходимы системы реального времени. Использование средств, обеспечивающих режим реального времени, значительно повышает уровень сложности разработки программ, что приводит к увеличению стоимости программного обеспечения.

Большой объем непосредственно адресуемого адресного пространства оперативной памяти 16-разрядных систем (часто превышающих 512К) обеспечивает реализацию новых, до сих пор неиспользованных возможностей. Кроме одновременного обслуживания нескольких пользователей, многозадачного режима и режима реального времени, эти системы обеспечивают высокий уровень защиты информации с помощью таких средств, как использование паролей и автоматическое обнаружение ошибок.

Система UNIX

С недавнего времени внимание специалистов оказалось обращенным к операционной системе UNIX для 16-разрядных микропроцессоров, созданной примерно в то же самое время,

что и CP/M. Эта система была разработана фирмой Bell Laboratories специально для серии мини-компьютеров PDP-11 фирмы Digital Equipment Corporation, но с недавнего времени она приспособлена для систем на базе широко распространенных микропроцессоров, включая 68000 и Z8000. Кроме того, эта система используется даже в системах на базе микропроцессора Z80. Система UNIX обеспечивает очень большие возможности, причем некоторые из них до недавнего времени были доступны только в больших и дорогих операционных системах, обслуживающих универсальные вычислительные машины.

Введение в систему UNIX

Система UNIX представляет собой операционную систему, обслуживающую одновременно нескольких пользователей и, кроме того, включающую утилиты и ассемблер для базового микропроцессора, а также транслятор с языка C, разработанный фирмой Bell Laboratories. Система обеспечивает получение реентерабельных программных кодов, раздельное размещение в памяти команд и данных, средства повышения безопасности данных, возможность обработки времени счета программ, управление взаимодействием нескольких программ и мультиплексный режим ввода-вывода для высокоскоростного обслуживания интенсивного потока данных между периферийными устройствами.

Несмотря на столь мощные возможности, система UNIX имеет ряд недостатков. Во-первых, она намного дороже большинства 16-разрядных операционных систем (но при этом фирма Bell Laboratories бесплатно поставляет эту систему учебным заведениям). Система UNIX была создана главным образом с ориентацией на управление выполнением программ, что также можно считать слабой стороной этой системы. Поскольку интерфейс с пользователем в этой системе достаточно сложен, использование ее требует определенного опыта программирования.

Один из недостатков системы UNIX заключается в отсутствии средств, ограничивающих доступ к файлам. Попытка обращения к одному и тому же файлу нескольких пользователей одновременно, может привести к тяжелым последствиям. При этом средства обнаружения ошибок слишком примитивны. Сбой в вычислительной системе часто приводит к порче больших объемов информации, даже если она записана на диске.

Система XENIX

Фирма Microsoft при участии фирмы Bell Laboratories разработала операционную систему XENIX для 16-разрядных

микропроцессоров, являющуюся развитием операционной системы UNIX. В системе XENIX были устранены некоторые недостатки операционной системы UNIX. В частности, уменьшен размер памяти для размещения системы, а также введены более мощные средства обработки ошибок. В отличие от системы UNIX операционная система XENIX рассчитана на работу одного пользователя в многозадачном режиме. Интерфейс этой системы с пользователем намного проще и позволяет работать неквалифицированным программистам. Поэтому эту систему можно использовать для решения коммерческих задач.

28.9. ЗАКЛЮЧЕНИЕ

Свойства выпускаемых микрокомпьютеров постоянно изменяются. Кроме того, растет число требований, зависящих от области применения каждого конкретного компьютера. Поэтому дальнейшее развитие операционных систем для микрокомпьютеров в реально обозримом будущем не остановится. Большое число уже существующих вычислительных систем на базе 8-разрядных микропроцессоров, а также большой объем разработанного для них программного обеспечения, предназначенного для функционирования под управлением таких систем, как CP/M, гарантирует долгую жизнь уже существующим стандартам операционных систем. Выпускаются новые версии существующих операционных систем, повышается их уровень, но требования совместимости с прежними вычислительными системами накладывает ограничения, препятствующие внесению в них коренных изменений.

Современный уровень технологии вызвал появление перспективных операционных систем для 8-разрядных микрокомпьютеров, и такие системы будут разрабатываться и в дальнейшем.

К настоящему времени начал увеличиваться выпуск операционных систем для 16-разрядных микропроцессоров. Особенности аппаратного обеспечения 16-разрядных микрокомпьютеров, а также различия в требованиях пользователей этих компьютеров скорее всего приведут к созданию ряда стандартных операционных систем, предназначенных для определенных целей, например разработки программного обеспечения, решения задач коммерческого характера и т. д. Процесс эволюции операционных систем для микрокомпьютеров только лишь начался.

ВЫВОД ЗВУКОВОЙ ИНФОРМАЦИИ: ВОСПРОИЗВЕДЕНИЕ МУЗЫКИ И РЕЧИ¹⁾

Л. Хоэнштейн

29.1. ВВЕДЕНИЕ

Назначение любого выходного устройства ЭВМ независимо от принципа действия состоит в том, чтобы выдавать пользователю информацию в привычной и удобной для его восприятия форме. Как правило, современное периферийное оборудование обеспечивает вывод либо текстовых материалов, либо графических изображений. Однако в последнее время начинают появляться и такие устройства, в которых предусмотрены средства для воспроизведения звуковой информации. Подобные приборы можно разделить на три основные группы:

1. Устройства для выдачи предварительно записанных сообщений.

2. Синтезаторы музыки и шумовых эффектов.

3. Синтезаторы речи.

Разумеется, с научной точки зрения наибольший интерес представляет разработка аппаратуры, позволяющей синтезировать человеческую речь. С этим тесно соприкасается, хотя и имеет свои особенности, проблема создания средств распознавания речи.

29.2. ЗВУКОВЫЕ ОТВЕТЧИКИ

Приборами этого типа оснащаются системы словесного ответа, называемые также системами автоматического оповещения. В таких системах для хранения речевой информации используются не аналоговые сигналы, как, например, в накопителях на магнитной ленте, а цифровые коды, которые заносятся в постоянную память компьютера, изготовленную на интегральных схемах. Применение постоянной памяти, обеспечивающей прямой доступ к данным, позволяет устранить задержки, необходимые для перемещения носителя в устройствах с лентопротяжными механизмами.

¹⁾ Adapted from Computer Peripherals for Minicomputers, Microprocessors, and Personal Computers, by Louis Hohenstein. Copyright © 1980. Used by permission of McGraw-Hill, Inc. All rights reserved.

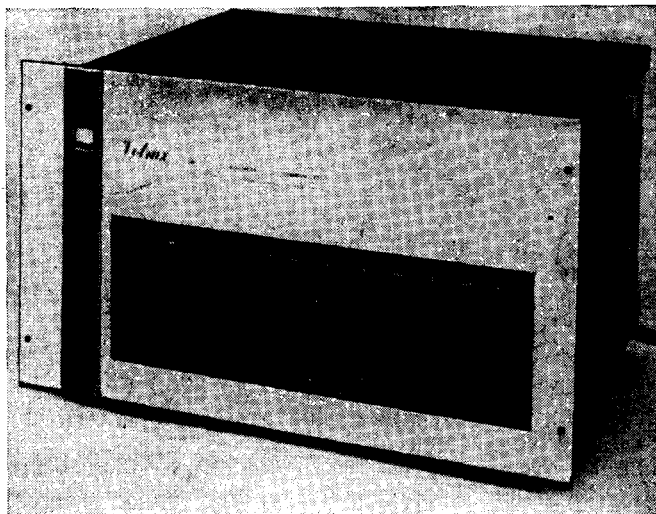


Рис. 29.1. Звуковой ответчик с цифровой памятью, рассчитанный на воспроизведение записанной речи в течение 128 с (отделение Votrax фирмы Federal Screw Works).

Появление звуковых ответчиков с цифровой памятью было вызвано потребностями предприятий телефонной связи, нуждавшихся в аппаратуре, с помощью которой можно было бы по выбору в зависимости от сочетания сигналов, приходящих по телефонным линиям, воспроизводить различные слова, номера или фразы. Например, нужно сообщить абоненту, что набранный им номер изменен на другой, и продиктовать ему этот новый номер. Впоследствии звуковые ответчики стали внедряться в торговле, промышленности и военном деле.

На рис. 29.1 приведена фотография распространенной модели звукового ответчика, оформленного в виде стандартного модуля ЭВМ. В этом модуле помещаются микропроцессор, память, содержащая управляющую программу, память с закодированной звуковой информацией, таймер реального времени, а также интерфейс последовательного обмена данными типа RS232C для сопряжения с центральным компьютером. Память, емкость которой рассчитана на непрерывное звучание в течение 128 с, обеспечивает прямой доступ к любой из речевых единиц длительностью по 0,5 с, или индивидуальную выборку до 256 сообщений переменной длины. Если потребуется увеличить продолжительность звучания (количество сообщений), к системе можно подключить дополнительные модули.

Запись слов диктора осуществляется путем перевода электрических сигналов, поступающих из микрофона, в цифровые коды. Речь, преобразованная таким способом в цифровую форму, заносится в программируемое постоянное запоминающее устройство. Для выборки информации достаточно опросить массив ячеек, в которых хранится закодированное слово или фраза. Их содержимое вновь преобразуется в непрерывные электрические сигналы, которые затем усиливаются до уровня, необходимого для их воспроизведения через телефон или громкоговоритель.

Серийно выпускаемые звуковые ответчики дают на выходе спектр частот, вполне достаточный для отчетливого восприятия произносимых слов. Например, устройство, показанное на рис. 29.1, имеет полосу частот от 50 до 3000 Гц. Это дает возможность добиться звучания, достаточно близкого по тембру к записанному голосу диктора.

Звуковой ответчик, изображенный на рис. 29.1, может одновременно обслуживать до 64 линий связи. Дополнительно предусмотрен специальный режим, в который устройство переходит автоматически при отключении центральной ЭВМ. В этом режиме, отвечая на поступающие запросы, устройство выдает сообщения типа «Система не работает. Позвоните, пожалуйста, по телефону 829-1988. До свидания».

29.3. СИНТЕЗАТОРЫ МУЗЫКИ И ШУМОВЫХ ЭФФЕКТОВ

Процесс синтеза музыки и шумовых эффектов построен на совершенно ином принципе, нежели работа звуковых ответчиков. Действительно, последние способны воспроизводить лишь наборы предварительно записанных слов (или другие сочетания звуков), в то время как синтезаторы, управляемые программистом и/или ЭВМ, формируют звуки из некоторых элементарных составляющих (обычно это чистые музыкальные тона). Таким образом, искусственная мелодия не является простым повторением ранее выполненной записи, но создается заново путем комбинирования элементарных звуковых единиц. Специализированные музыкальные синтезаторы получили достаточно широкое распространение. Ими пользуются современные эстрадные группы, для того чтобы придать ритмичность или обогатить звучание основной мелодии, исполняемой музыкантом (с этой же целью синтезаторами оснащаются и бытовые электроорганы); их используют композиторы и исполнители, экспериментирующие в области новых музыкальных форм, а также любители, увлекающиеся составлением программ для персональных компьютеров (в настоящее время выпускается несколько недорогих моделей музыкальных синтезаторов, сопрягаемых с

персональными ЭВМ); наконец, их применяют в машинных играх для имитации звуков, издаваемых при полете ракет, стрельбе из оружия, ударах по мячу и создания самых разнообразных шумовых эффектов.

Простейшие синтезаторы имеют обычно один звуковой канал, по которому в соответствии с заложенной программой могут попеременно генерироваться до 128 или 256 дискретных гармоник. Типичным примером может служить модель, снабженная усилителем с полосой частот от 107 до 27 500 Гц, который вырабатывает синусоидальные сигналы 256 тонов с программируемой длительностью звучания.

Более совершенные музыкальные синтезаторы способны воспроизводить несколько различных тонов одновременно. Кроме того, в их памяти хранятся специальные таблицы, задающие форму генерируемых сигналов. Эти таблицы используются для придания мелодии определенного тембра. Меняя содержащуюся в них цифровую информацию, можно добиться желаемой окраски звуков. Для подобных синтезаторов разработано множество программ-компоновщиков. Пользователь может и сам запрограммировать музыкальную партитуру с указанием размера, темпа исполнения, пауз и повторов, силы и тембра звучания нот в диапазоне, охватывающем до четырех и даже шести октав. Как правило, мощные синтезаторы имеют два или четыре канала усиления.

29.4. СИНТЕЗАТОРЫ РЕЧИ

Подобно музыкальным синтезаторам, которые в отличие от звуковых ответчиков не воспроизводят записанную музыку, а komponуют мелодии из элементарных гармонических составляющих с заданными частотами и фазами, машинные синтезаторы речи строят осмысленные фразы из элементарных речевых единиц, называемых фонемами. Ниже будет описана система, в которой используется набор из 53 команд генерации фонем и четырех команд, задающих уровень удара. Располагая столь обширным выбором фонемных и ударных команд, пользователь может запрограммировать любое слово, образовав соответствующую последовательность фонем.

Уже сейчас для лиц с ослабленным зрением выпускаются «говорящие» терминалы и калькуляторы; ими удобно пользоваться и для начального обучения детей, которые не только смогут посмотреть, как пишутся слова, но и услышать, как они произносятся. Следует ожидать, что по мере совершенствования синтезаторов речи они станут широко применяться в различных сферах промышленности и торговли.

Типовой синтезатор включает три основных элемента:

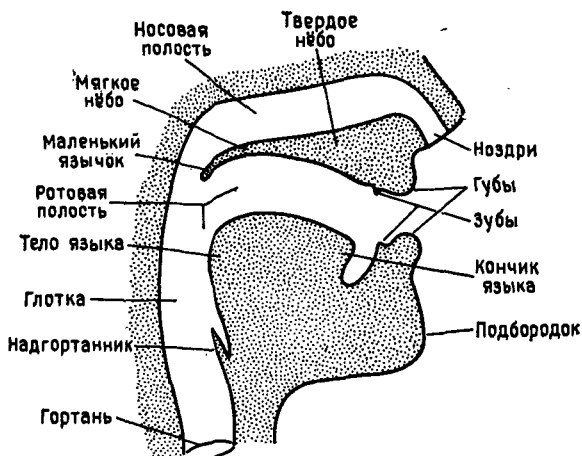


Рис. 29.2. Голосовой аппарат человека.

1. Электронный аналог голосового аппарата человека.

2. Программу, которая за счет изменения регулируемых параметров голосового аппарата формирует требуемую последовательность звуков.

3. Блок сопряжения, задающий параметры голосового аппарата.

В устройствах, выпускаемых различными фирмами, процедуры синтеза речи строятся практически по одной и той же схеме; далее на конкретных примерах демонстрируются ее основные принципы.

Устройство голосового аппарата

Принято считать, что голосовой аппарат человека начинается с гортани (голосовой щели), расположенной в средней части горла, и, разветвляясь, простирается вплоть до ротовой полости, ограниченной губами, а с другой стороны, — до носовой полости, заканчивающейся ноздрями (рис. 29.2). Общая длина акустического тракта составляет от 160 до 190 мм. Воздух, выходящий равномерным потоком из легких, действует на голосовые связки, возбуждая их колебания. Когда гортань сужается, эти колебания передаются всему голосовому аппарату, который ведет себя как полый резонатор, закрытый со стороны гортани. Ротовая полость с раскрытыми губами играет роль основной камеры резонатора, а носовая полость, приоткрываемая в нужные моменты, — вспомогательной (рис. 29.3).

Вся деятельность голосового аппарата управляется мускулами, которые регулируют расход воздуха, натяжение голосо-

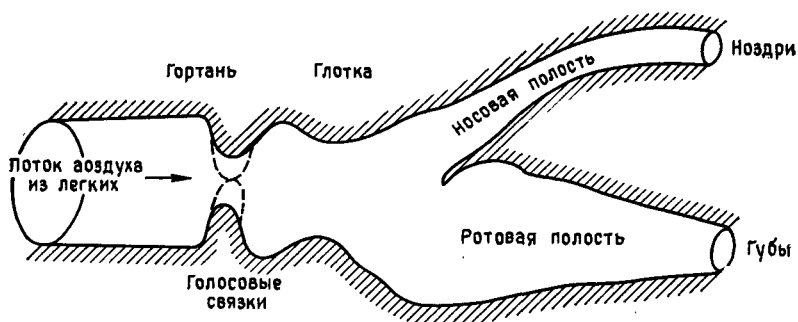


Рис. 29.3. Схематическая модель голосового аппарата, включающая источник звука (голосовые связки, колеблющиеся под напором воздуха), а также ротовую и носовую полости, которые играют роль резонаторов.

вых связок, форму ротовой полости, положение языка и т. д. В силу этого процесс произнесения любого слова, представляющего набор особым образом чередующихся звуков, можно отобразить последовательностью мускульных сокращений. Они определяют формирование элементарных звуков, или фонем, из которых и складывается слово.

В описываемой ниже типовой системе синтеза искусственной речи реализовано 48 согласных и гласных фонем, 7 фонем, специфичных для английского языка, 4 знака препинания и 4 уровня ударения. Фонемы далеко не всегда соответствуют буквам алфавита. В принципе буквы можно было бы объединить в группы, каждая из которых однозначно задавала некоторую фонему, однако правила перевода букв в фонемы оказываются чрезвычайно сложными. Дело в том, что одни и те же буквы в зависимости от конструкции слов, в которых они используются, дают разные звуки. Более того, и сами слова в различных географических районах часто звучат очень непохоже.

Можно полагать, что у человека процесс чтения текста вслух должен протекать следующим образом. При взгляде на слово, записанное в виде некоторой совокупности букв, человек произвольно формирует образ, представляющий определенный набор характерных для его речи звуков (фонем). Эти фонемы преобразуются далее в мускульные сокращения, управляющие движением легких и всех элементов голосового аппарата. В результате издаются звуки, служащие средством человеческого общения, которые мы называем речью. Практически та же последовательность действий реализуется и в синтезаторе речи, работающем под управлением компьютера,

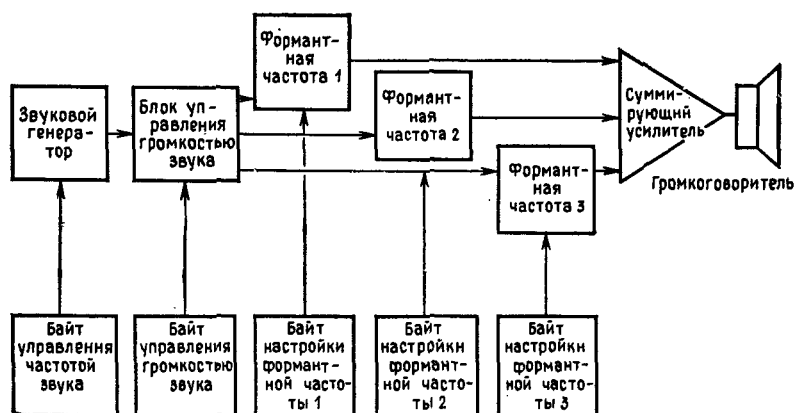


Рис. 29.4. Функциональная схема электронного устройства, воспроизводящего гласные звуки за счет суммирования формантных частот.

Основные элементы синтезатора речи

Формирование гласных звуков. Структура электронного синтезатора речи в упрощенном виде повторяет устройство голосового аппарата человека, который можно рассматривать как закрытую с одного конца систему акустических резонаторов длиной около 170 мм. Движение воздуха в акустической системе такой протяженности порождает стоячие волны с частотами порядка 500, 1500 и 2500 Гц. Эти три резонансные частоты, именуемые **формантными** (их обозначают F_1 , F_2 и F_3), по существу являются наиболее выраженными в соответствующих полосах резонансных частот, которые определяют тембровую окраску голоса при произнесении гласных звуков. В голосовом аппарате распределение колебаний вблизи указанных формантных частот регулируется мускулами, управляющими изменением формы горла и рта. В электронном синтезаторе речи аналогом подобного механизма служат три усилителя, снабженные полосовыми фильтрами, параметры которых могут изменяться в зависимости от цифровых сигналов, вырабатываемых ЭВМ (рис 29.4). На каждый из этих усилителей подается напряжение переменной амплитуды, колеблющееся с частотой, которая соответствует одной из формант, возбуждаемых в гортани.

Описанная часть синтезатора предназначена для воспроизведения гласных *a*, *e*, *i*, *o*, *u*. Каждому из этих звуков отвечает определенное сочетание формантных частот F_1 , F_2 и F_3 . Роль мускулов, регулирующих акустические характеристики голосового аппарата, здесь играют усилители с полосовыми фильтрами, настройка которых на требуемые частоты осуществ-

вляется в соответствии с содержанием трех управляющих 8-разрядных байтов.

В качестве примера на рис. 29.5 представлена диаграмма типичного распределения характеристических частот для нескольких похожих по звучанию слов, содержащих гласные *a*, *e*, *i* и *o*. Их называют установившимися, поскольку формантные частоты F_1 и F_2 мало меняются в процессе произнесения этих гласных. Что же касается третьей частоты F_3 , то она во всех гласных практически одинакова и поэтому на рисунке не показана. Таким образом, для того чтобы сгенерировать один из перечисленных гласных звуков с помощью аппаратуры, изображенной на рис. 29.4, достаточно включить синтезатор, предварительно застав в специально выделенные ячейки памяти три 8-разрядных управляющих байта, задающие требуемые формантные частоты F_1 , F_2 и F_3 .

Содержимое управляющих байтов поступает в синтезатор из памяти через шину данных периодически, с тактом 10 или 20 мс (в разных моделях синтезаторов). Этот такт характеризует **предельный темп речи**, т. е. в данном случае ту скорость, с которой могут меняться параметры, определяющие воспроизводимые звуки.

При генерации гласных в словах, указанных на рис. 29.5, управляющие параметры могут оставаться неизменными, поскольку эти гласные являются установившимися. Однако во многих словах гласные встречаются не поодиночке, а группами (как, например, в слове *boy*). Произносятся подобные слова, голос плавно переходит с одной гласной на другую; такие сочетания называются **дифтонгами**. Траектории изменения формантных частот в процессе воспроизведения дифтонгов показаны для нескольких английских слов на рис. 29.6. Длительность произнесения дифтонга составляет от 150 до 250 мс, и за это время в синтезаторе, работающем с тактом 10 мс, формантные частоты успевают смениться (за счет засылки новых управляющих байтов) 15—25 раз. При этом не просто задаются начальные и конечные значения частот, соответствующие двум гласным

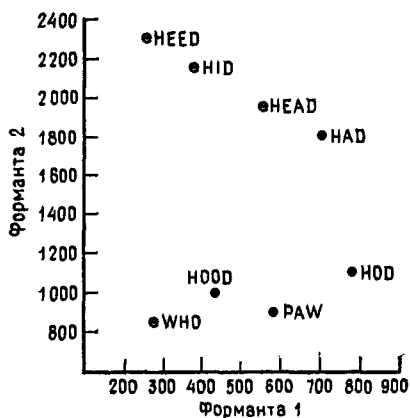


Рис. 29.5. Соотношения формантных частот для нескольких английских слов, содержащих установившиеся гласные звуки (с разрешения фирмы Byte Publications, Inc.).

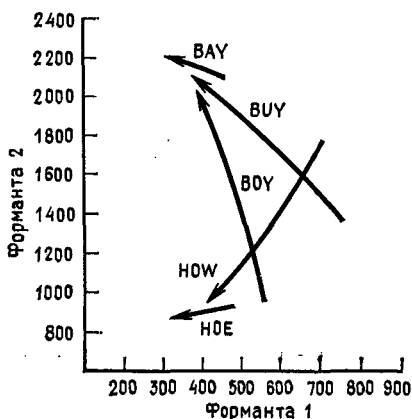


Рис. 29.6. Дифтонги, характеризующиеся плавным изменением формантных частот при переходе от одного гласного звука к другому (с разрешения фирмы Byte Publications, Inc.).

дифтонга, но выдерживается определенный характер их изменения во времени.

Формирование согласных звуков. Для того чтобы синтезатор речи мог воспроизводить не только гласные, но и согласные звуки, в его состав, помимо функциональных элементов, представленных на рис. 29.4, должны быть включены еще три: источник шума, резонатор сонант и резонатор фрикативных согласных.

С помощью источника шума имитируется придыхание, т. е. мгновенное размыкание губ, сопровождающееся энергичным выбросом воздуха, которое возникает при произнесении взрывных согласных

p, t, k, b, d и *g*. В частности, согласный звук *k* образуется путем быстрого приоткрывания задней части ротовой полости с одновременным выпуском мощной струи воздуха через губы, после чего происходит немедленный переход к следующему за *k* гласному звуку (например, *ka* в слове *kay*).

В одной из распространенных моделей речевого синтезатора источником придыхания служат шумы, возникающие при лавинном пробое зинеровского диода. Амплитуда шума придыхания определяется содержимым байта, управляющего восемью резисторами (в совокупности они образуют цифро-аналоговый преобразователь). Далее шум придыхания требуемой амплитуды смешивается в усилителе-микшере с несущей частотой, вырабатываемой звуковым генератором, после чего подается на входы трех резонаторов формантных частот (рис. 29.7).

Шум придыхания используется, кроме того, для воспроизведения фрикативных согласных звуков *sh, zh, s, z, f, v* и *th* (все они характеризуются частотами, лежащими в диапазоне 2500—8000 Гц). Фрикативные согласные формируются в резонаторе фрикативных звуков, работающем в полосе частот от 2500—8000 Гц). Фрикативные согласные формируются в резонаторе шума придыхания. Выход резонатора регулируется как по амплитуде, так и по частоте. Воспроизведение требуемого фрикативного звука осуществляется путем настройки резонатора на определенную частоту;

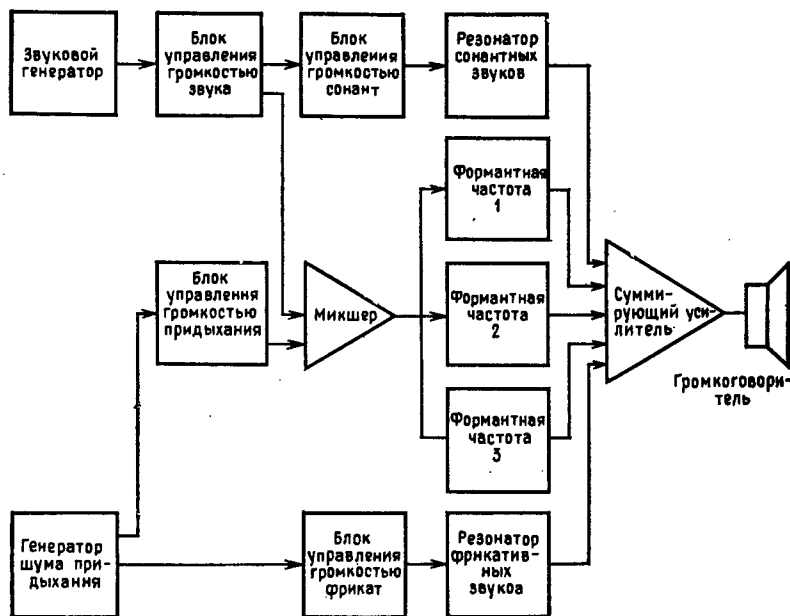


Рис. 29.7. Полная функциональная схема электронного аналога голосового аппарата человека.

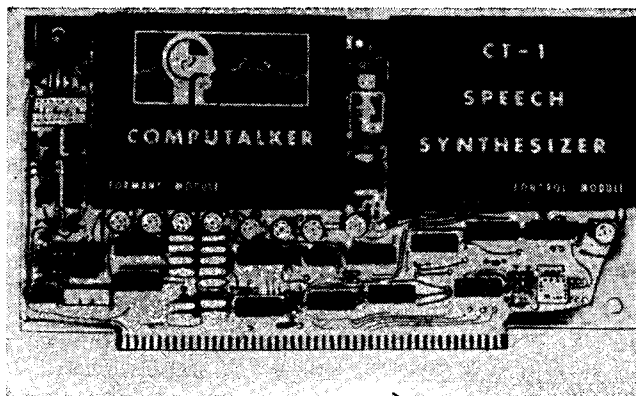


Рис. 29.8. Синтезатор речи, смонтированный на печатной плате (производство фирмы Computalker Consultants).

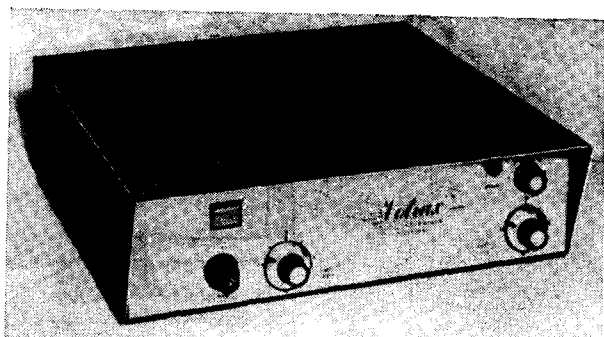


Рис. 29.9. Электронный синтезатор речи типа VS-6, выпускаемый фирмой Votrax (отделение фирмы Federal Screw Works).

Носовые согласные m , n и ng образуются в резонаторе сонант, вырабатывающем узкополосный сигнал с частотой около 1400 Гц. Амплитуда сигнала задается резисторами, которые регулируются соответствующим управляющим байтом. Электронные компоненты, реализующие все описанные функции (показаны в виде укрупненных модулей на рис. 29.7), смонтированы на печатной плате, фотография которой приведена на рис. 29.8. Очень важно, что эта плата допускает непосредственное сопряжение с информационными шинами ряда микрокомпьютеров. Для того чтобы получить практически полный аналог голосового аппарата человека, остается лишь подключить к плате громкоговоритель.

На рис. 29.9 показан еще один синтезатор речи, оформленный в виде отдельного модуля. Дополнительно к нему могут подсоединяться несколько различных устройств сопряжения, предназначенных для передачи данных в последовательном и параллельном режимах. С помощью регуляторов, смонтированных на передней панели синтезатора, устанавливаются желаемый темп речи, высота голоса и уровень громкости.

Регулируемые параметры электронного синтезатора

Подобно тому как мускулы управляют действиями голосового аппарата, в серийных синтезаторах речи процесс образования гласных и согласных звуков управляется последовательностью изменения регулируемых параметров (обычно их число равно девяти). Каждый параметр (например, громкость источника звука) имеет 264 дискретных уровня, задаваемых сопротивлениями восьми резисторов, зависящими от содержания соответствующего 8-разрядного байта. Совокупность из девяти байтов (плюс байт, отвечающий за включение/выключе-

ние устройства) полностью определяют значения регулируемых параметров в синтезаторе, фотография которого приведена на рис. 29.9. Перечислим эти параметры:

амплитуда сигнала на выходе звукового генератора;

частота сигнала на выходе звукового генератора;

амплитуда шума придыхания;

амплитуда сонантных звуков;

амплитуда фрикативных звуков;

формантная частота 1;

формантная частота 2;

формантная частота 3;

частота сигнала на выходе резонатора фрикативных звуков.

Управляющие байты выбираются из оперативной памяти ЭВМ по адресам, задаваемым содержимым 4-разрядного счетчика (корпус его интегральной схемы можно увидеть в правом нижнем углу платы, показанной на рис. 29.8).

Каждый набор управляющих байтов, последовательно передаваемых в синтезатор через информационную шину компьютера, определяет фрагмент речи длительностью 10 мс. Меняя каждые 10 мс содержимое ячеек, используемых для хранения этих байтов, можно строить последовательности регулируемых параметров, с помощью которых формируются не только отдельные гласные и согласные звуки, но и целые слова и предложения. Например, для воспроизведения слова «hello» (здравствуйте), на произнесение которого затрачивается около 0,7 с, необходимо образовать 70 наборов из девяти регулируемых параметров, задающих отдельные фрагменты речи, каждый из которых генерируется в течение 10 мс.

Существуют два подхода к определению последовательностей регулируемых параметров. Первый из них основан на применении фонетических правил, второй — на результатах анализа спектрально-энергетического состава реально произносимых слов в различных частотных диапазонах.

Расчет регулируемых параметров

по данным анализа спектрально-энергетического состава слов

Описываемый метод позволяет добиться правильного и весьма естественного звучания речи, однако он отличается крайней трудоемкостью. Параметры, управляющие работой синтезатора, рассчитываются по предварительно полученным спектрально-энергетическим характеристикам, меняющимся в процессе словообразования.

Один из способов определения указанных характеристик предусматривает непосредственное преобразование записанного на пленку речевого сигнала в цифровую форму с частотой

10 000 Гц. Эту цифровую информацию обрабатывают с помощью специальных математических алгоритмов и получают в результате оценки значений несущей частоты голоса, а также изменений формантных частот.

Принципиально иной способ оценивания характеристических частот основан не на математическом анализе последовательности оцифрованных значений речевого сигнала, а на использовании набора полосовых фильтров, с помощью которых выделяются несущая и формантные частоты. Сигналы, вырабатываемые фильтрами, также подвергаются оцифровке с частотой 10 000 Гц, после чего с периодом 1 мс вычисляются их средние амплитуды. Полученные частотные характеристики преобразуются в значения регулируемых параметров F_1 , F_2 и F_3 , которые затем масштабируются таким образом, чтобы они укладывались в диапазоне целых чисел от 0 до 255, записываемых в 8-разрядных управляющих байтах.

После завершения всех процедур частотного анализа, выполняемых на ЭВМ, приходится вручную осуществлять окончательную корректировку процессов изменения формантных частот. Цель этой операции состоит в удалении выбросов, обусловленных случайными помехами, и сглаживании кривых на тех участках, где имеются разрывы. Наконец, на завершающем этапе добавляются сигналы, имитирующие шум придыхания и фрикативные согласные звуки.

Представление о трудоемкости описанного метода дают следующие цифры: на обработку записи одного произнесенного вслух предложения для последующего расчета последовательности формантных частот затрачивается от 10 до 100 ч машинного времени (разумеется, этот показатель зависит и от длины предложения, и от производительности компьютера). И кроме того, необходима еще ручная доводка регулируемых параметров, чтобы обеспечить требуемое качество воспроизведения речи.

Тем не менее в тех случаях, когда нужно добиться максимально естественного звучания коротких, но часто повторяемых фраз, этот метод оказывается наиболее эффективным. Еще одним достоинством такого подхода, в котором машинный анализ дополняется ручной обработкой, является его независимость от языка, благодаря чему он может с успехом применяться для синтеза слов на любом языке или диалекте.

Определение регулируемых параметров по фонетическим правилам

При синтезе слов английского языка на основе правил фонетики девять регулируемых параметров, управляющих работой синтезатора, формируются в соответствии с буквенными

Таблица 29.1. Коды, используемые для записи звуков английской речи

Код	Пример употребления	Код	Пример употребления
Согласные звуки			
P	pie	SH	shy
T	tie	ZH	visiou
K	key	L	lie
B	by	W	we
D	die	R	rye
G	guy	Y	you
M	my	HH	high
N	nigh	CH	chime
NX	hang	JH	jive
F	fie	WH	why
V	vie	EL	battle
TH	thigh	EM	bottom
DH	thy	EN	botton
S	sigh	Q	задержка звука
Z	zoo		в гортани
Гласные звуки			
IY	heed	ER	herd
IH	hid	AH	Hud
EY	hayed	AY	hide
EH	head	AW	how
AE	had	OY	boy
AA	hod	AX	about
AO	hawed	IX	David
OW	hoed	OH	core
UH	hood	UX	too
UW	who'd		
Другие обозначения			
KX	soo (K перед гласным звуком)	DX	pity (T между двумя гласными)
GX	goo (G перед гласным звуком)	YX	окончание дифтонга
RX	card (R после гласной)	WX	окончание дифтонга
LX	kill (L после гласной)		
Знаки ударения			
0	безударный звук	2	ударение третьего уровня
1	максимальный уровень ударения	4	ударение четвертого уровня
2	ударение второго уровня	5	ударение пятого уровня
Знаки пунктуации			
—	промежуток между словами	.	падающая высота звука
,	пауза (тишина)	?	нарастающая высота звука

AN — амплитуда сонант
 FF — частота резонатора фри-
 кативных согласных
 AF — амплитуда фрикативных
 согласных
 AH — амплитуда шума приды-
 хания
 F1 — формантная частота 1
 F3 — формантная частота 3

 F2 — формантная частота 2
 F0 — несущая частота звука

 AV — амплитуда звукового
 генератора

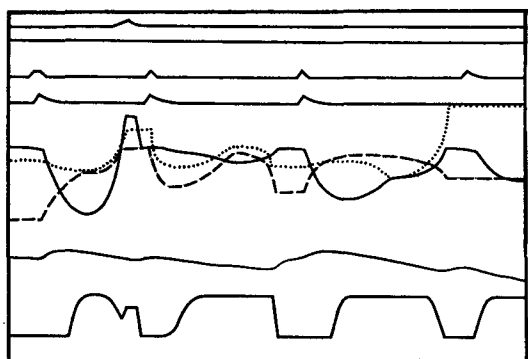


Рис. 29.10. Параметры, определяющие произносимые звуки, рассчитываются согласно фонетическим правилам; каждому фонемному коду ставятся в соответствие значения девяти регулируемых параметров. Установившиеся значения, отвечающие парам соседних кодов, связываются гладкими параболическими кривыми, которые строятся согласно таблицам, задающим эти установившиеся значения и длительность их звучания (предоставлено фирмой Computalker Associates).

обозначениями, которыми кодируются 53 фонемы английской речи (табл. 29.1). Фраза, записанная с помощью таких кодов, весьма отдаленно напоминает английский текст и больше похожа на набор произносимых по слогам слов. В синтезаторах, выпускаемых разными фирмами, используются различные системы кодирования фонем. Например, предложение «I can talk pretty well» (Я неплохо умею разговаривать) в синтезаторе фирмы Computalker Consultants кодируется так: «AY2 KAEN TOA1LK PRITIY WEHL». Пожалуй, не всякий, глядя на этот текст, поймет, что перед ним знакомые английские слова. Цифры в такой записи обозначают силу ударения. А вот как выглядит фраза, содержащая специальные термины, «Please, fix my fricatives» (пожалуйста, научите меня правильно выговаривать фрикативные согласные): «PLIY1Z FIN3KS MAY FRIN2KAN2IHVZ». Приведем еще один пример, показывающий, как синтезатор на словах может пояснять вычисления, выполняемые компьютером: «...TAY2MZ TEN2ENN TUN DNAX FON4RTH PAY3WER» — это закодированная запись фрагмента фразы «...Times ten to the fourth power» («...на десять в четвертой степени»).

Слова и фразы, представленные последовательностями закодированных фонем, обрабатываются программой-компоновщиком, которая формирует девятки управляющих байтов, действуя согласно алгоритму, учитывающему не только сами фонемы, но и их взаимовлияние.

В итоге каждой фонеме ставится в соответствие набор регулируемых параметров и задается время ее звучания. Для получения плавного перехода между соседними фонемами программа-компоновщик связывает значения присвоенных им параметров гладкими кривыми, имеющими форму парабол (рис. 29.10).

В качестве иллюстрации процесса словообразования ниже для 13 английских слов приведены фонетические записи, которые строятся по-разному в зависимости от того, какая гласная фонема помещается между начальной согласной фонемой НН и завершающим согласным звуком D:

Фонетическое представление	Английское слово
NNIYD	heed
NNIHD	hid
NHEYD	hayed
NNEHD	head
NNAED	had
NNAAD	had
NNAOD	hawed
NNOWD	hoed
NNUHD	hood
NNUWD	who'd
NHERD	herd
NNAND	hud
NNAYD	hide

Для каждой из трех фонем, образующих перечисленные слова,— согласной НН, заданного гласного звука и согласной D — с помощью специального алгоритма рассчитываются значения управляющих байтов и связывающие их переходные кривые; по ним устанавливаются величины регулируемых параметров, сменяемых каждые 10 мс. Результатом работы алгоритма является скомпонованная последовательность управляющих параметров, определяющая то слово, которое будет впоследствии воспроизведено синтезатором речи.

Ф. Коперда

30.1. ВВЕДЕНИЕ

Распознавание речи — эта важная область исследований, ведущихся уже свыше двадцати лет, предоставляет широкие возможности для применения микропроцессорной техники. Первые экспериментальные системы строились на базе мощных ЭВМ, обладающих высоким быстродействием, необходимым для выполнения трудоемких вычислений. Со временем на смену этим машинам пришли мини-ЭВМ; наряду с ростом их производительности совершенствовались также алгоритмические и программные средства. А теперь уже имеются микропроцессоры, по своим характеристикам сравнимые с мини-компьютерами. И вполне реальным становится создание систем распознавания речи на основе этих новых микроэлектронных приборов.

Можно назвать по меньшей мере две потенциальные сферы применения систем распознавания речи: 1) ввод данных в ЭВМ «с голоса», что очень удобно для неспециалистов в области вычислительной техники; 2) полуавтоматические системы контроля качества, в которых операторы, работая вручную, лишены возможности отвлекаться на ввод данных с клавиатуры.

Высокая стоимость большинства серийных устройств распознавания и весьма бедный запас воспринимаемых ими слов не способствовали широкому внедрению их в практику. Почти все они имеют и еще один изъян: будущий пользователь должен в течение некоторого времени «обучать» устройство, чтобы оно «свыклось» с особенностями его произношения. В идеале система распознавания должна не только обладать обширным словарем, но и воспринимать речь любого пользователя.

В этой главе описываются принципы построения экспериментальной системы, разработанной автором в порядке личной инициативы. Пока она реализована далеко не полностью, но

¹⁾ Adapted from Microprocessor Applications Handbook, edited by David F. Stout. Copyright © 1982. Used by permission of McGraw-Hill, Inc. All rights reserved.

некоторые важные компоненты завершены и успешно функционируют. Результаты их пробной эксплуатации продемонстрировали перспективность системы, и в настоящее время ведется работа по созданию остальных ее частей. Основная идея этого проекта состоит в использовании информации о форме колебаний, присутствующих в речевом сигнале. Частично эта информация была заимствована из научных изданий, частично получена экспериментально путем наблюдения процессов с помощью осциллографов и графопостроителей.

30.2. ЦЕЛИ ПРОЕКТА

Звуковые колебания, возникающие при произнесении слов, представляют сложные сигналы, форма которых сильно изменяется в зависимости от анатомического строения голосового аппарата, эмоционального состояния говорящего и его индивидуальных привычек. Основным элементом механизма словообразования являются голосовые связки, колебания их и порождают звуки, прежде всего гласные. В формировании большинства звуков участвуют также язык, нос и губы, которые в той или иной степени препятствуют свободному току воздуха (примером могут служить звуки [s] и [f]). Характер произношения элементарных звуков в значительной мере определяется формой и размерами полостей рта и носа, а также других воздушных каналов, имеющих в носоглотке. Эти анатомические факторы и служат причиной того, что так непохоже звучат голоса не только разных людей, но часто и одного человека в разное время.

Очевидно, система распознавания речи может считаться эффективной только в том случае, когда она мало чувствительна к вариациям произношения и способна функционировать независимо от индивидуальных особенностей говорящего. Указанные вариации определяются изменениями частоты, амплитуды и длительности колебаний. Следовательно, система должна быть спроектирована так, чтобы эти параметры подвергались нормализации и на их основе вырабатывались обобщенные характеристики, не зависящие от абсолютных значений частоты, амплитуды и продолжительности сигнала. Таким образом, данный подход предполагает использование относительных величин, не зависящих от абсолютных значений измеряемых и рассчитываемых показателей.

В ходе проектирования преследовались еще две цели: сделать систему нечувствительной к посторонним звукам и добиться того, чтобы она распознавала непрерывную речь. Нечувствительность к посторонним звукам — это необходимая предпосылка использования системы в реальных условиях, когда неизбеж-

но присутствие внешних шумов. Такая ситуация характерна, в частности, для любого производства, где множество шумов порождается работающим оборудованием, людьми и т. п. В такой обстановке, естественно, нельзя рассчитывать на создание идеальных условий, в которых отсутствовали бы всякие посторонние звуки.

Требование того, чтобы система распознавала непрерывную речь, также вызвано практическими соображениями. Некоторые системы достаточно точно идентифицируют отдельные слова, которые выговариваются с отчетливыми паузами. Что же касается непрерывной речи, представляющей последовательность слов, произносимых естественным голосом, то достоверность ее распознавания оказывается значительно ниже. Объясняется это в первую очередь тем, что система вынуждена определять моменты, когда кончается одно слово и начинается другое. Поскольку во многих приложениях приходится иметь дело именно с непрерывной естественной речью, система, вынуждающая пользователя выговаривать слова особым образом, по отдельности, не может быть признана удовлетворительной.

Наконец, еще одна цель проекта — создать такую систему, которую можно было бы построить на микропроцессорах. Данное требование делает невозможным использование многих разработанных ранее методов, так как для их реализации необходимы вычислительные ресурсы, далеко превосходящие возможности современной микропроцессорной техники. Например, один из распространенных подходов к распознаванию речи основан на применении быстрого преобразования Фурье (БПФ), с помощью которого получают описание сигнала в частотной области. Однако для выполнения БПФ в реальном масштабе времени быстродействия микропроцессора совершенно недостаточно, ввиду чего в рассматриваемой системе переход к частотному описанию не предусмотрен. По той же причине (недостаточной производительности) в системе используется не один, а несколько микропроцессоров, за счет чего удастся достигнуть необходимой суммарной вычислительной мощности. Это лишь два примера, показывающие, что при проектировании микропроцессорных систем распознавания речи во многих случаях приходится искать нестандартные решения.

30.3. ОБЩАЯ ХАРАКТЕРИСТИКА СИСТЕМЫ

В большинстве известных систем процесс распознавания речи осуществляется в шесть этапов: 1) преобразование входного непрерывного сигнала в цифровую форму путем его квантования; 2) сжатие информации, т. е. выделение из нее тех по-

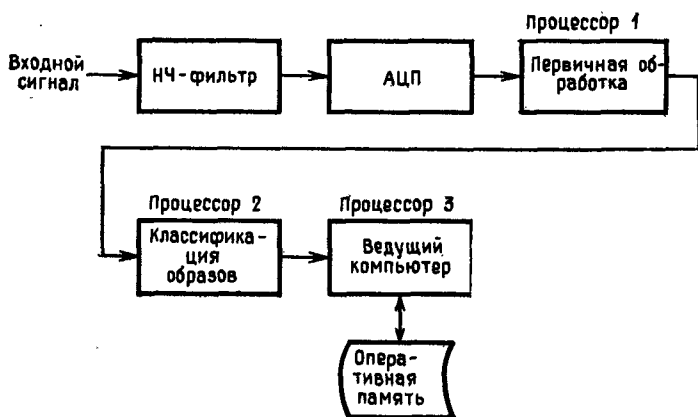


Рис. 30.1. Структура конвейерной системы распознавания речи.

казателей, которые подлежат дальнейшей обработке; 3) определение границ слова; 4) разделение слова на элементарные образы; 5) классификация образов; 6) сравнение последовательности образов с эталонными данными, содержащимися в системном словаре.

При наличии мощного процессора все эти шесть задач можно было бы решать параллельно. Но, ориентируясь на использование микропроцессоров, целесообразнее строить систему по конвейерной схеме, включающей несколько последовательно соединенных процессоров. При такой архитектуре первый процессор решает некоторую часть перечисленных задач и передает результаты обработки во второй процессор. После этого первый процессор возобновляет свои операции над следующей порцией информации, в то время как второй выполняет дальнейшую обработку полученных им данных. Общее число действующих в системе процессоров определяется целым рядом факторов: сложностью выполняемых операций на различных этапах обработки, быстродействием применяемых микропроцессоров, ограничениями по стоимости, требованиями по точности распознавания и т. д.

На рис. 30.1 представлена блок-схема системы конвейерного типа, обсуждаемой в этой главе. Кроме входных электронных цепей, в ее состав входят три микропроцессора, выполняющие самостоятельные функции. На входе системы стоит микрофонный предусилитель, имеющий резко падающую характеристику на частотах выше 8 кГц. За ним следует аналого-цифровой преобразователь (АЦП), который переводит непрерывный звуковой сигнал в последовательность цифровых кодов. С аналого-цифрового преобразователя они поступают в процессор 1

(входной микропроцессор), в задачу которого входит выделение полезного сигнала на фоне помех. Кроме того, он фиксирует и заносит в память пиковые значения сигнала (локальные максимумы и минимумы), а также интервалы времени между пиками. Благодаря этому значительно уменьшается объем данных, участвующих в дальнейшей обработке. Сжатая таким способом информация передается в процессор 2, где определяется, может ли она рассматриваться как звуковой образ. Если алгоритм дает положительный ответ, то формируется набор параметров, зависящих от типа образа. Процессор 3, который в конвейерной системе играет роль ведущего, на основе этих параметров выносит заключение о том, какое слово было произнесено, а затем выполняет соответствующее запрограммированное действие (выводит слово на печать, выдает команду исполнительному устройству и т. д.). Нетрудно видеть, что конвейерная структура системы отражает последовательность этапов обработки информации в процессе распознавания речи.

Возможен и другой подход, позволяющий за счет введения дополнительной аппаратуры несколько сократить объем вычислительных операций. Он основан на применении полосовых фильтров, с помощью которых речевой сигнал разделяется на ряд частотных диапазонов, а затем преобразуется в цифровую форму. Далее полученная информация о спектральном составе сигнала может непосредственно использоваться для определения коэффициентов разложения в ряд Фурье или для расчета других частотных характеристик. При этом все особенности речевого сигнала воспроизводятся и в его частотном описании. Однако этот подход имеет свои недостатки: аналоговые фильтры недешевы, а для их цифровой реализации требуется весьма высокое быстродействие, которое пока нельзя обеспечить с помощью микропроцессорных средств.

30.4. ПЕРВИЧНАЯ ОБРАБОТКА РЕЧЕВЫХ СИГНАЛОВ

Показанная на рис. 30.2 осциллограмма колебаний, возникающих при произнесении слова *ship*, дает представление о сигналах, обрабатываемых системой распознавания речи. На первом этапе обработки аналоговый сигнал подвергается квантованию, и его значения в тактовые моменты времени преобразуются в стандартные цифровые коды, которые далее могут участвовать в любых вычислительных операциях, выполняемых компьютером. Проблемы, связанные с преобразованием сигналов, — это выбор частоты квантования и фильтрация помех.

Аналого-цифровые преобразователи (АЦП). Перевод аналоговых сигналов в цифровые коды осуществляется с помощью АЦП, которые по своему действию практически ничем не отли-

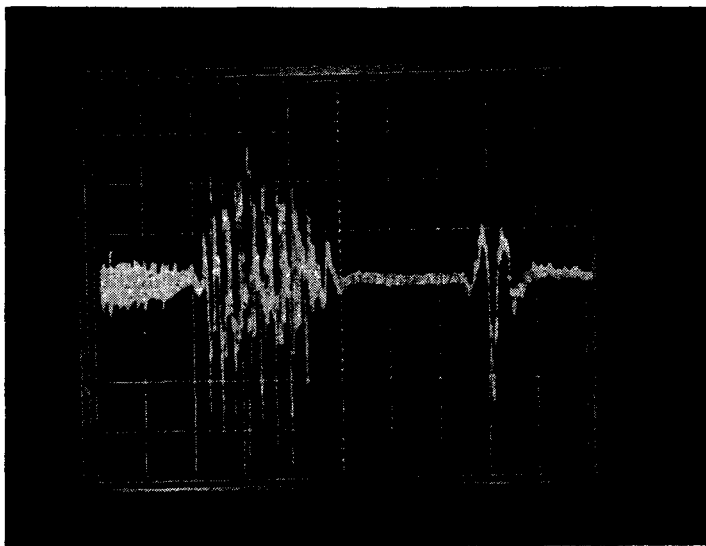


Рис. 30.2. Образ, соответствующий слову ship.

чаются от цифровых вольтметров, управляемых ЭВМ. Было предложено много разнообразных схем построения АЦП, в том числе основанных на последовательном приближении, преобразовании напряжения в частоту, интегрировании разнополярных сигналов. Для оцифровки речевых сигналов целесообразно использовать АЦП, работающие по принципу последовательного приближения, поскольку такие преобразователи выпускаются в виде относительно недорогих интегральных микросхем и обладают достаточно высоким быстродействием.

В аналого-цифровом преобразователе с последовательным приближением входной сигнал сравнивается с рядом эталонных напряжений, по которым последовательно определяются цифры его двоичного представления. Сначала аналоговый сигнал сравнивается с эталонным напряжением, равным половине рабочего диапазона АЦП. Если входное напряжение превосходит эталонное, то в старший разряд выходного регистра заносится двоичная единица. Затем эталонное напряжение увеличивается наполовину, и вновь производится сравнение. Если входной сигнал по-прежнему больше эталонного, единица заносится во второй разряд. Эта процедура продолжается до тех пор, пока не будет образовано полное двоичное представление входного сигнала, занимающее весь регистр. Описанный процесс сравнения напряжений происходит внутри микросхемы АЦП,

так что процессор в нем не участвует; он лишь запускает преобразователь, а по завершении операции считывает данные.

Максимальная разрядность современных АЦП в микросхемном исполнении доходит до двенадцати (это соответствует разрешению в 1/4096), а преобразователь с 8 разрядами — это уже широко распространенный прибор, имеющий вполне доступную цену. Испытания разработанной системы показали, что 8-разрядный АЦП обеспечивает приемлемую точность результатов.

Требования к быстродействию АЦП определяются двумя факторами: частотой квантования входного сигнала и необходимым разрешением по времени. В следующем разделе будет показано, что частота квантования должна составлять не менее 6 кГц. Что же касается нужного разрешения, то для его получения такт работы АЦП должен быть во много раз меньше периода в 160 мкс, соответствующего частоте квантования 6 кГц. Отметим, что в обсуждаемой системе был использован АЦП, имеющий длительность преобразования 20 мкс, т. е. способный работать с частотой 50 кГц.

Существуют и другие способы кодирования аналоговых сигналов, в частности полиномиальная экстраполяция (ПЭ). На практике применяются несколько вариантов полиномиальной экстраполяции, в том числе линейная экстраполяция с постоянным и переменным опорным напряжением. Некоторые из перечисленных методов преобразования непрерывных сигналов обсуждаются в последующих разделах.

Выбор частоты квантования. Речевой сигнал занимает диапазон частот, нижняя граница которого чуть меньше 100 Гц, а верхняя достигает 8 кГц и более. Согласно критерию Найквиста, произвольный периодический сигнал можно полностью описать, имея как минимум $2f$ его дискретных значений, где f — частота высшей гармоники непрерывного сигнала⁴⁾. Следовательно, для точного воспроизведения речи частота квантования должна составлять не менее 16 кГц, а такт квантования — не более 60 мкс. Однако эти цифры справедливы лишь при условии, что все частоты речевого сигнала не превышают 8 кГц; в действительности же он содержит и более высокочастотные составляющие. Чтобы избежать искажений, вызванных неправильным выбором такта квантования (эффект «транспонирования частот»), целесообразно ограничить полосу входного сигнала, воспользовавшись низкочастотным фильтром с частотой среза порядка 8 кГц. Частота квантования 50 кГц, с которой работает описываемая система, обеспечивает адекватное воспроизведение сигналов в полосе шириной до

⁴⁾ Данный вывод следует из теоремы Котельникова. — *Прим. перев.*

25 кГц. С учетом характеристик используемого микрофонного предусилителя этого вполне достаточно, чтобы гарантировать отсутствие искажений при переводе речевого сигнала в цифровую форму.

Подавление шумов. При распознавании речи, произносимой в естественной обстановке, почти всегда приходится сталкиваться с разного рода внешними шумами. Можно уменьшить влияние подобных помех, применяя направленные микрофоны, снабженные дополнительными шумопоглощающими экранами. С помощью микрофонов такого типа удается достигнуть существенного снижения уровня внешних шумов по сравнению с амплитудой полезного речевого сигнала.

Для борьбы с помехами, обусловленными работой различного механического оборудования (например, электродвигателей), успешно применяются также высокочастотные фильтры, подавляющие сигналы в области малых частот. Эти фильтры, как правило, строятся на аналоговых элементах, но могут быть реализованы и в цифровой форме на микропроцессоре. Что же касается случайных посторонних звуков продолжительностью порядка нескольких миллисекунд, то для их выявления и устранения целесообразно использовать именно микропроцессор.

30.5. СЖАТИЕ ИНФОРМАЦИИ

Для запоминания всей информации, получаемой в результате квантования речевого сигнала со скоростью 16 000 тактов в секунду, понадобилась бы память чрезвычайно большой емкости. Кроме того, для последующей обработки такого количества данных просто не хватит производительности микропроцессора. Это означает, что необходимо найти способ более экономного хранения или сократить объем информации, не допуская в то же время таких ее потерь, которые затруднили бы дальнейший процесс распознавания речи.

Один из возможных подходов к проблеме сжатия информации связан с использованием специальных схем кодирования, в частности, линейных экстраполяторов, особенно удобных для преобразования непрерывных речевых сигналов. Преобразователи такого типа выдают оцифрованные значения разности амплитуд между двумя тактовыми моментами времени. Если эта разность меньше некоторой пороговой величины, информация в память не заносится, благодаря чему удается существенно сократить объем данных, описывающих медленно изменяющиеся сигналы. Физически линейный экстраполятор представляет собой интегратор, на который подается знакопеременный ступенчатый сигнал.

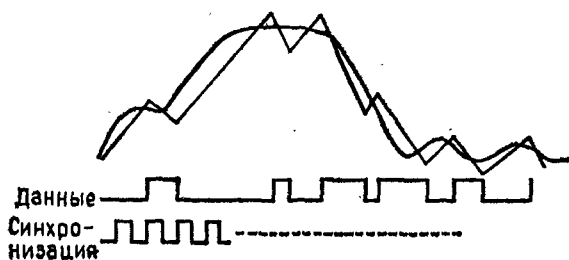


Рис. 30.3. Линейная экстраполяция непрерывного сигнала.

Как видно из рис. 30.3, начиная с момента $t = 0$ на выходе интегратора формируется линейно нарастающий сигнал (интеграл от постоянной). В каждом такте он сравнивается с входным сигналом и если оказывается больше последнего, то знак опорного напряжения, подаваемого на интегратор, меняется на противоположный. После этого напряжение на выходе интегратора начинает равномерно уменьшаться (второй линейный отрезок на рис. 30.3). Когда выход интегратора становится меньше входного сигнала, знак опорного напряжения вновь изменяется. Таким образом, интегратор генерирует кусочно-линейный сигнал, который с некоторым запаздыванием «отслеживает» колебания входного. При этом для получения цифрового описания непрерывного сигнала достаточно только фиксировать моменты времени, когда происходит смена знака опорного напряжения на входе интегратора. Фактически в память заносится либо количество тактовых импульсов, приходящееся на каждый линейный отрезок, либо двоичные единицы при изменении знака и нули в остальных тактах.

Точность, с которой интегратор отслеживает входной сигнал, определяется величиной опорного напряжения и частотой квантования. При правильном выборе этих параметров и не слишком высоких требованиях к точности воспроизведения сигнала (разрешению по уровню) можно весьма существенно сократить объем запоминаемой информации — для речевого сигнала он составляет всего около 2 кбайт в секунду.

Описанная схема может быть усовершенствована за счет введения нескольких опорных напряжений, подаваемых на интегратор (рис. 30.4). Когда входной сигнал быстро изменяется (т. е. идет с резким наклоном и соответственно имеет большую производную), используется максимальное опорное напряжение, благодаря чему выход интегратора быстрее успевает «догнать» входной сигнал. Если же наклон невелик, то прикладывается малое напряжение, что позволяет избежать больших динамических ошибок. При таком способе преобразо-

вания необходимо фиксировать не только моменты переключения, но и уровни опорного напряжения.

В обоих рассмотренных способах экстраполяции предполагается, что в микропроцессоре текущие значения входного сигнала восстанавливаются путем численного интегрирования записанных в память данных. Однако реализованный в системе алгоритм распознавания (он описывается ниже) построен так, что для его работы нужны лишь отношения последовательных значений речевого сигнала. По этой причине использовать методы кодирования, основанные на линейной экстраполяции, нерационально, поскольку численное интегрирование, хотя и позволяет экономить память, связано в то же время с чрезмерно трудоемкими вычислениями.

В системе был применен несколько иной подход; его основная идея заключалась в оперативном анализе информации с целью отбора только тех данных, которые играют определяющую роль в процессе дальнейшей обработки. Как только из АЦП поступает очередное оцифрованное значение, оно сравнивается с предыдущим, с тем чтобы определить, не изменился ли в течение такта знак производной входного сигнала. Если изменение имело место, замеренная амплитуда фиксируется в памяти. Одновременно записывается длительность интервала, отсчитываемого с момента предшествующей смены знака. В некоторой степени это напоминает принцип действия пикового детектора, который аппроксимирует входной сигнал пилообразным, с переменной амплитудой и различными промежутками между пиками (рис. 30.5).

Такой метод дает возможность раздельно анализировать амплитудные и частотные характеристики речевого сигнала, поскольку вторые непосредственно определяются интервалами между максимумами и минимумами. Этот способ отбора данных иногда называют амплитудно-частотной сепарацией. Строго говоря, величины, обратные интервалам, нельзя считать истинными частотами, т. е. параметрами синусоид в математическом смысле. Тем не менее они дают важную информацию о спектральном составе речевого сигнала. Чтобы дать представление о достижимой степени сжатия данных, достаточно сказать, что описания большинства коротких слов (таких, как ship) уместаются в памяти совсем небольшого объема — всего около 500 байт,



Рис. 30.4. Линейная экстраполяция с изменяемым опорным напряжением.

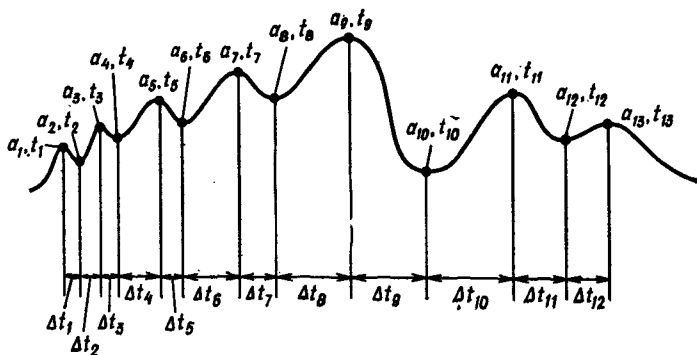


Рис. 30.5. Амплитудно-частотная сепарация.

При использовании амплитудно-частотной сепарации требуется высокая точность измерения временных интервалов. Экспериментально было установлено, что разрешение по времени должно быть не хуже 80 мкс (это соответствует частоте квантования 12 500 Гц). Как уже упоминалось, в разработанной системе такт составляет 20 мкс, что гарантирует хорошее разрешение.

30.6. ОПРЕДЕЛЕНИЕ ГРАНИЦ СЛОВ

Целью следующей стадии обработки информации в системе распознавания речи является разграничение отдельных слов. Прежде всего необходимо определить, действительно ли произнесено слово, и если это так, то надлежит выяснить, где оно начинается и кончается. На первый взгляд это — элементарная задача, однако она осложняется присутствием разного рода помех и электрических наводок. Имеется по меньшей мере четыре показателя, руководствуясь которыми можно выделить речь на фоне шума, — это амплитуда, частота, продолжительность и форма колебаний. Каждый из них по отдельности не всегда служит надежным критерием, но в совокупности они обеспечивают высокую вероятность разделения слов.

Из-за того, что применяется несколько показателей, процедура определения границ слов распадается на ряд этапов. Наиболее просто, еще на этапе ввода, идентифицируются границы изолированных, четко произносимых слов. Что же касается непрерывной естественной речи, когда слова быстро следуют друг за другом и выговариваются не всегда отчетливо, то в этом случае выявление границ приходится откладывать до более поздних этапов процесса распознавания.

Настройка пороговой амплитуды. Достаточно эффективным средством определения начала и конца последовательности звуков, квалифицируемых как произнесенное слово, служит механизм настройки пороговой амплитуды. С его помощью установленное пороговое напряжение постепенно понижается до тех пор, пока не появится сигнал, превосходящий этот уровень. Если та часть системы, которая осуществляет классификацию образов, расценивает поступившие данные как фоновый шум, порог повышается. Спустя некоторое время, короткое по сравнению с продолжительностью звучания любого слова, уровень вновь начинает снижаться, и вся последовательность действий повторяется. При таком подходе система сама настраивается на прием реальной речи, и в то же время память ее не перегружается ложной информацией.

Человеческая речь обладает интересной особенностью, которая создает определенные трудности для выявления концов слов. Когда слово заканчивается, возбуждение голосовых связок спадает, но они еще некоторое время продолжают вибрировать. Ослабление колебаний речевого сигнала в это время носит весьма неравномерный характер, и спады амплитуды могут ошибочно интерпретироваться как повторяющиеся паузы. В результате бывает трудно установить, где же именно кончается слово. Как оказалось, и эта проблема успешно решается с использованием механизма настройки пороговой амплитуды. Если интервал между сменами знака производной сигнала превосходит 10 мс, то пороговый уровень повышается, а частота квантования уменьшается до 1000 Гц (это соответствует такту 1 мс). После небольшой задержки исходный уровень восстанавливается, но квантователь продолжает работать с тактом 1 мс. Как только амплитуда сигнала превысит пороговое значение, частота квантования вновь доводится до своей максимальной величины.

Существует еще одно явление, приводящее к ошибкам в определении конца слова. Если слово содержит несколько фонем (так называются элементарные звуковые единицы речи), переходы между ними достигают 10 мс и более. Для дальнейшей обработки очень важно правильно выявлять эти переходы. Когда пауза длится свыше 10 мс, это может означать не только конец слова, но и конец фонемы. Действительную причину паузы можно установить, снижая частоту квантования по истечении указанного периода до 1 кГц и восстанавливая одновременно прежний пороговый уровень. Если спустя 10 мс появляется сигнал с амплитудой выше этого уровня, считается, что встретился конец фонемы. Если же эта и последующие проверки не выявляют наличия сигнала, фиксируется окончание слова. Цифра 10 мс была выбрана, исходя из результатов пред-

варительных экспериментов, и не исключено, что в дальнейшем она будет корректироваться.

Частота сигнала. Частота входного сигнала, которая оценивается по промежуткам между переключениями знака его производной, может служить критерием для отделения истинного речевого сигнала от фоновых помех. Во многих типах шумов (в частности, вызываемых работой механического оборудования) преобладают сравнительно низкочастотные колебания. Для их подавления используются аналоговые фильтры, но те же функции с успехом может выполнить и микропроцессор. Он оценивает частоту входного сигнала и проверяет, не лежит ли она ниже границы в 300 Гц. Если для сжатия информации используется метод амплитудно-частотной сепарации, микропроцессор не принимает входные данные до тех пор, пока интервалы между сменами знака производной превышают 1,5 мс, т. е. половину периода синусоиды с частотой 300 Гц. При этом конец слова целесообразно идентифицировать по исчезновению сигнала с частотой свыше 300 Гц. В рамках данного подхода микропроцессор играет роль быстродействующего компаратора и одновременно фильтра низких частот. Даже при высокой интенсивности фона удастся снизить вероятность ложного срабатывания до приемлемого уровня.

Форма колебаний. Еще один способ выявления речи на фоне шума основывается на анализе формы колебаний входного сигнала. В большинстве случаев шумы с частотами выше 300 Гц имеют случайный характер и отличаются малой продолжительностью. Результаты предварительных экспериментов позволяют считать, что всякий изолированный звук продолжительностью менее 100 мс нельзя расценивать как начало произносимого слова, ввиду чего все записанные в этот период данные следует игнорировать.

Слитная речь. Проведенные эксперименты, в которых слова произносились с паузами или по крайней мере выговаривались очень отчетливо, продемонстрировали эффективность описанной методики разграничения слов и подавления шумов. Однако можно полагать, что в практических ситуациях при обработке разговорной речи с присущими ей плавными переходами между словами результаты могут оказаться не столь блестящими. Например, сочетание *you all* произносится жителями южных штатов США как слитное *y'all*, и поэтому в процессе распознавания его приходится рассматривать как единое слово. Поскольку предложенная система реализована пока не полностью, трудно судить о ее эффективности при распознавании слитной речи. Вероятнее всего придется либо использовать иной метод разграничения слов, либо анализировать слитные словосочетания целиком.

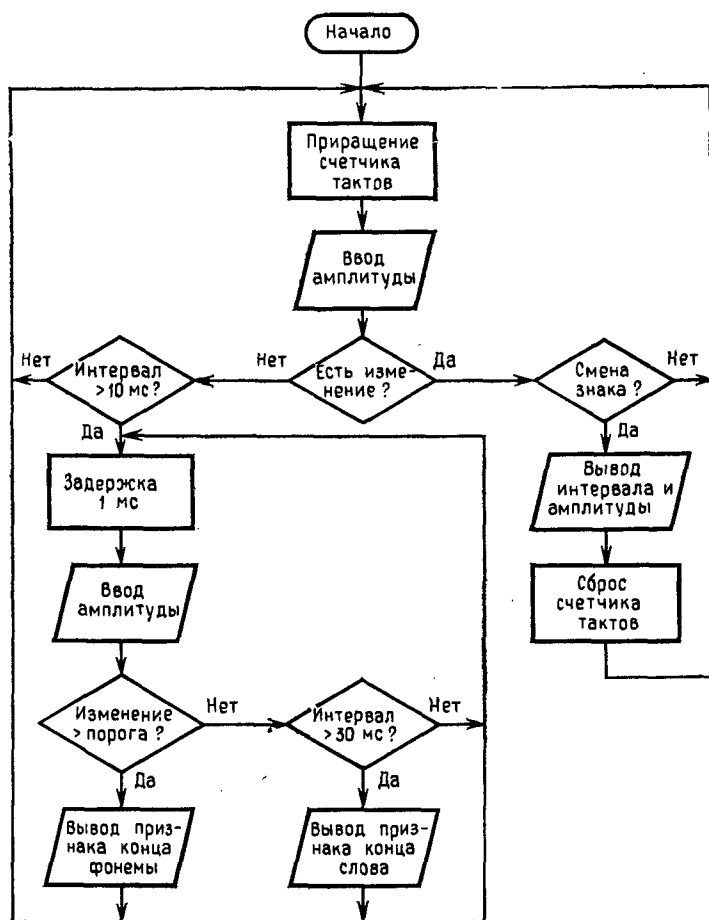


Рис. 30.6. Схема алгоритма на основе амплитудно-частотной сепарации.

Процедура выделения слов. Метод амплитудно-частотной сепарации, обеспечивающий одновременное устранение помех, сжатие информации и определение границ слов, был положен в основу алгоритма первичной обработки речевого сигнала. На рис. 30.6 приведена схема этого алгоритма; как показали эксперименты, он позволяет достаточно точно воспроизводить форму процессов на входе системы. Программа алгоритма занимает всего около 100 строк на языке ассемблера для стандартного 8-разрядного микропроцессора.

Верхними блоками на рис. 30.6 представлена та часть алгоритма, в которой анализируется производная входного сигнала. Для достаточно точной фиксации моментов смены

знака эта часть должна выполняться не более чем за 80 мкс (но желательно быстрее). Из схемы видно, что выбор такта квантования не сказывается на объеме данных, описывающих форму входных колебаний. Действительно, данные заносятся в память только в тех случаях, когда наклон сигнала меняется на противоположный. Поэтому увеличение частоты квантования приводит лишь к повышению точности определения промежутков между максимумами и минимумами. Когда происходит смена знака, значение амплитуды сигнала либо записывается в память, либо направляется в следующий микропроцессор. Одновременно запоминается интервал между соседними переключениями знака, и цикл повторяется далее до тех пор, пока не произойдет очередное изменение наклона.

Блоки, показанные в нижней левой части рис. 30.6, отображают часть процедуры, предназначенную для выявления пауз (между словами или в начале некоторых согласных), во время которых данные не передаются для дальнейшей обработки. Введение в такие периоды пониженной частоты квантования и использование переменной пороговой амплитуды помогает правильно идентифицировать шумы и уменьшить вероятность ошибок в определении конца слова. Согласно алгоритму, данные не заносятся в память, пока не будет сделан вывод о том, что действительно произносится слово. Благодаря этому из обработки исключается значительная часть бесполезной информации. Применение метода амплитудно-частотной сепарации позволяет существенно уменьшить объем запоминаемых данных, например для простых коротких слов типа *one* требуется всего от 256 до 512 байт оперативной памяти.

Метод амплитудно-частотной сепарации обладает еще одним достоинством — он дает возможность использовать очень простую аппаратуру при воспроизведении звуков речи. Для этого требуется лишь цифро-аналоговый преобразователь (ЦАП), снабженный простейшим апериодическим фильтром. Данные, полученные с помощью описанного выше алгоритма, последовательно пересылаются в ЦАП, на выходе которого формируется пилообразный сигнал, аппроксимирующий первичные звуковые колебания. Для улучшения качества звучания в микропроцессоре может быть реализована процедура, с помощью которой каждая пара точек связывается соответствующим отрезком синусоиды.

30.7. ОБЪЯСНЕНИЕ НЕКОТОРЫХ ЛИНГВИСТИЧЕСКИХ ТЕРМИНОВ

В последующих разделах употребляется ряд лингвистических терминов, с которыми большинство читателей незнакомо. Возможно, приведенные здесь определения специалисты сочтут

недостаточно строгими, заметим, впрочем, что среди самих лингвистов нет единодушия относительно толкования некоторых терминов. Однако мы не стремимся к полной строгости и хотим лишь дать общее представление о тех понятиях, которые используются в оставшейся части главы. Усвоить эти сведения из лингвистики лучше всего помогут многочисленные примеры. Ознакомившись с данным разделом, читатель, кроме того, поймет, что в распознавании речи основную роль играет анализ звуков, а не буквенных изображений слов.

Морфемы. Морфемами называются элементарные семантические единицы языка. В английском языке каждое слово содержит по меньшей мере одну морфему. Многие морфемы являются самостоятельными словами, например *pin* и *fish*. Очень часто конструкция, которую мы именуем словом, получается в результате объединения нескольких морфем. Например, каждая из морфем *up*, *pin* и *ing* имеет собственную семантику, но *up*, обозначающая отрицание, и *ing* — совершаемое действие, порознь не употребляются.

Фонемы. Фонема — это элементарная звуковая единица человеческой речи. Например, в слове *cat* содержится три фонемы, причем каждая из них отображается отдельной буквой. Это далеко не всегда так, поскольку многие фонемы записываются двумя буквами, подобно звуку, представленному сочетанием *oi* в слове *house*. В то же время некоторые буквы могут порождать несколько фонем. В частности, букве *a* соответствует три различные фонемы, примеры звучания которых дают слова *cat*, *father* и *take*. Наконец, иногда буква вообще не произносится, так обстоит, например, с немой *e* в слове *time*. Разного рода несоответствия между написанием и произношением (неоднозначность графемно-фонемной связи) весьма характерны для английской речи и даже вынудили лингвистов разработать специальный фонетический алфавит, в котором для каждой фонемы предусмотрен свой символ.

Согласные. К согласным относятся все буквы английского алфавита, кроме *a*, *e*, *i*, *o* и *u*. Как правило, одиночной согласной соответствует единственная фонема. Но есть и исключения, так, например, буква *c* собственного звука не имеет и произносится иногда подобно *k* (*cat*), а иногда как *s* (*cent*). В сочетании с другими некоторыми согласными могут порождать и большее число фонем.

Гласные. В английском алфавите основными гласными являются пять букв — *a*, *e*, *i*, *o* и *u*. Но иногда в качестве гласной выступает и *y*. Всякая морфема в английском языке содержит по меньшей мере одну гласную. Гласные дают множество нарушений однозначной графемно-фонемной связи. Даже поодиночке гласные могут порождать несколько фонем, звучание

которых определяется написанием слов, т. е. теми буквами, которые окружают гласную. Например, а в словах рап и рапе произносится по-разному, поскольку последнее дополняется немой е.

На гласные очень часто воздействуют следующие за ними согласные, например буква о в слове boat меняет свое звучание в слове тоге из-за появления буквы г.

Диграфы. Диграфом называется сочетание двух гласных или двух согласных, порождающее всего одну фонему. Примерами могут служить sh в слове chair и ai в слове sail.

Дифтонги. Две рядом стоящие гласные объединяются в дифтонг, если обе они произносятся и звучат так, как не звучат в комбинациях с другими буквами. Дифтонгами являются, например, буквосочетания ou в слове out и oi в слове oil.

30.8. ВЫДЕЛЕНИЕ ЗВУКОВЫХ ОБРАЗОВ

Следующий шаг обработки информации в обсуждаемой системе относится уже к этапу анализа. Пока не совсем ясно, чем именно отличается осмысленная речь от других звуков, по каким характеристикам речевого сигнала человеческий мозг осуществляет распознавание слов. Тем не менее есть основания полагать, что в процессе распознавания важную роль играют звуковые образы, т. е. фрагменты сигнала, характеризующиеся определенными соотношениями амплитуд и частот. Если принять такую точку зрения, то всякую речь можно рассматривать как некоторое сочетание образов нескольких типов и переходов между этими образами.

Очень трудно четко сформулировать, что должно отличать образ, поскольку в речи могут наблюдаться сигналы самого разнообразного вида. Однако в первом приближении можно считать, что звуковой образ — это достаточно часто встречающийся и выделяющийся среди других набор колебаний особой формы.

Изучение звуковых колебаний, запечатленных на экране осциллографа или на графопостроителе, позволяет выделить три их особенности. Большинство звуков, произносимых без придыхания, как, например, |s|, дают колебания случайного характера, которые тем не менее обладают вполне определенными параметрами. В гласных звуках наблюдаются периодические процессы сложной структуры, вызываемые колебаниями голосовых связок. Взрывные согласные характеризуются короткой паузой и последующим скачкообразным возрастанием амплитуды сигнала. Разумеется, в реальных условиях «базовые» звуки произносятся с самыми разнообразными оттенками, которые определяются свойственными индивидууму движениями рта, языка и другими особенностями артикуляции. В результате

каждый образ оказывается весьма специфичной характеристикой, присущей только конкретному лицу.

Отсюда следует вывод, что в таком виде понятие образа в значительной степени теряет смысл и нуждается в каком-то обобщении.

Даже в пределах одной фонемы образ постепенно деформируется, а между соседними фонемами наблюдаются переходы, варьирующиеся от полного исчезновения звука до сложных процессов, в которых друг на друга накладываются начальный и конечный образы. Следовательно, первый этап анализа должен состоять в выявлении таких сравнительно устойчивых образов, которые в дальнейшем можно было бы подвергнуть классификации с целью распознавания произнесенного слова.

Входной информацией для этого этапа служат значения максимумов и минимумов звукового сигнала, а также интервалы между ними. Временные интервалы весьма удобно использовать для выделения участков случайного сигнала, а значения амплитуд — для восстановления формы колебаний. В то же время ни один из этих показателей по отдельности не является достаточно информативным, чтобы с его помощью можно было достоверно оценивать параметры, характеризующие образ.

Особенности образов. У случайных звуковых сигналов знак производной часто меняется, так как интервалы между переключениями обычно не превышают 160 мкс. Возможное присутствие взрывных согласных ($[b]$, $[d]$, $[g]$, $[p]$, $[t]$, $[k]$) определяется на этапе ввода данных или в процессе анализа образов по паузам, сопровождаемым скачкообразным возрастанием амплитуды. Если промежуток между резкими скачками амплитуды составляет порядка 10 мс, то с большой вероятностью можно полагать, что встретилась взрывная согласная.

Несколько более сложную задачу представляет анализ изменений амплитуды, связанных с периодическими колебаниями сигнала. В большинстве случаев колебания являются затухающими, т. е. амплитуда их в пределах образа постепенно уменьшается, как показано на рис. 30.7, а. Присутствие затухающих колебаний можно обнаружить, вычитая очередное пиковое значение из предыдущего и определяя, образуют ли модули вычисленных в нескольких точках разностей монотонно убывающую последовательность. Данные, получаемые в результате выполнения такого алгоритма, иллюстрируются на рис. 30.7, б.

Процедура идентификации образов. Весьма не просто с достаточной точностью определить, где начинается и где кончается образ. Даже самые простые слова, скажем *one*, содержат три-четыре типа образов, причем каждый из них встречается по 10—12 раз. Выявление таких одиночных групп представляет сложную проблему, что обусловлено наличием

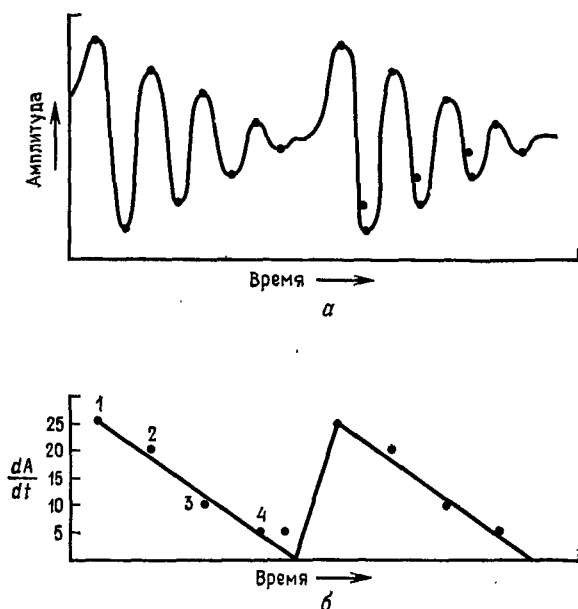


Рис. 30.7. Затухающие колебания (а); изменение модулей разности пиковых значений для затухающих колебаний (б).

значительных вариаций образов в рамках одного и того же типа, а также переходных процессов, предвещающих и завершающих каждый образ.

В обсуждаемой системе процедура определения начала и конца образа включает ранее описанные алгоритмы распознавания затухающих колебаний и сигналов случайного типа. Когда установлено, к какому из этих двух классов должен быть отнесен сигнал, считается, что зафиксировано начало образа. В тот момент, когда характер поступающей информации изменяется, фиксируется завершение текущего образа и начинается поиск следующего.

Рассмотрим теперь процедуру более подробно. После того как установлено, что произносится слово, программа начинает анализировать входные данные, проверяя, являются ли они случайными, о чем можно судить по коротким интервалам между переключениями наклона, не превышающими 160 мкс, или представляют затухающие колебания, на что указывает периодическое монотонное уменьшение модулей разностей пиковых значений. Если проверка подтверждает случайность данных на протяжении примерно 30 смен знака, поиск затухающих колебаний прекращается. В то же время если семь раз подряд

проверка дает отрицательный результат, делается вывод о завершении образа. Точно так же при обнаружении затухающих колебаний конец образа фиксируется после семи смен знака, которые не удовлетворяют проверке на колебательность.

Когда установлено, что встретился конец образа, определяется общее число смен знака производной в пределах образа. Если их меньше сорока, то считается, что вся записанная в это время информация представляет переходный процесс и исключается из последующего анализа.

Описанная процедура была построена на основе данных, полученных в ходе экспериментов и в результате теоретического изучения речевых сигналов, и как показали проведенные испытания, она обеспечивает достаточно высокую точность разделения слов на образы, которые используются на последующих этапах обработки. Различия в индивидуальных особенностях речи и даже изменения интонации у одного говорящего приводят к тому, что в одном и том же слове могут появляться заметно отличающиеся наборы образов. Проблема неоднозначности должна решаться на этапах классификации образов и распознавания слов.

30.9. ПАРАМЕТРЫ ОБРАЗОВ И ИХ АНАЛИЗ

На очередном этапе обработки, после того как слово разделено на образы, предстоит найти ряд числовых параметров, используемых в процедурах классификации и распознавания. Если описать образ очень детально, весьма маловероятно, что в том же слове встретится образ с совпадающими параметрами. С другой стороны, чрезмерно грубое описание затрудняет идентификацию звуков, в частности гласных. Для того чтобы система распознавания обладала малой чувствительностью к изменению манеры речи, она должна допускать определенные вариации образов.

Для описания речевых сигналов можно использовать различные наборы параметров. Некоторые из них требуют проведения сложных математических расчетов и соответственно значительных затрат процессорного времени или применения дополнительных аппаратных средств. Следовательно, для того чтобы система в целом была достаточно эффективной, необходимо соблюсти разумный компромисс между производительностью и стоимостью, с одной стороны, и точностью — с другой.

Типы образов. В описываемой системе процедура определения параметров образа включает два шага. На первом анализируются типы колебательных процессов, на втором формируется описание образа. Для каждого шага исходной информацией служат значения амплитуд и временных интервалов.

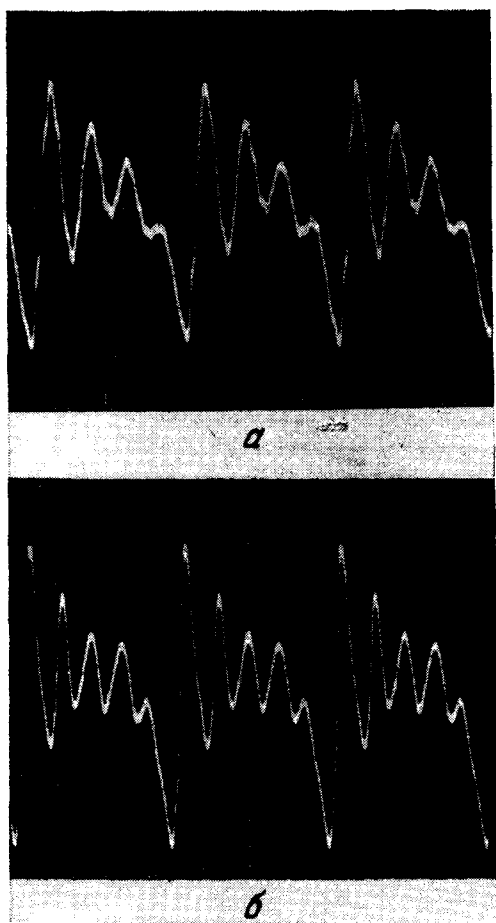


Рис. 30.8. Сбразы, соответствующие звукам $|a|$ (а) и $|o|$ (б).

Оцениваются изменения амплитуды пиков и промежутков между ними, причем большинство параметров вычисляется в виде отношений, с тем чтобы они не были связаны с конкретными размерностями измерений времени и уровня сигнала.

В настоящее время используется 18 параметров, 16 из которых объединены в четыре группы. Оставшиеся два параметра отличаются от остальных, первый из них определяется как сумма, второй — как интервал, усредненный по всему образу. Набор параметров описывает следующие характеристики образов:

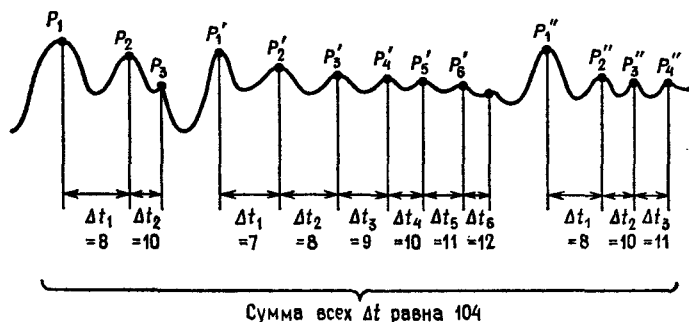


Рис. 30.9. Данные, используемые при анализе затухающих колебаний.

1. Затухающие колебания — анализ амплитуд (четыре отношения).

2. Затухающие колебания — анализ временных интервалов (четыре отношения).

3. Пиковые значения — анализ амплитуд (четыре отношения).

4. Пиковые значения — анализ временных интервалов (четыре отношения).

5. Суммарное число повторений.

6. Усредненный временной интервал.

Затухающие колебания — анализ амплитуд. Как уже говорилось, признаком затухающих колебаний считается монотонное убывание амплитуды трех или более следующих друг за другом пиков. Как правило, в образах подобного типа колебательные процессы несколько раз повторяются. При этом оказывается, что количество колебаний (пиковых значений) в каждом процессе можно связать с определенным гласным звуком. Присутствие повторяющихся колебаний, содержащих для разных гласных звуков различное количество пиков в каждом повторении, хорошо видно из социллограмм, показанных на рис. 30.8.

По данным, относящимся к одному образу, устанавливается наличие повторений, и, если они имеются, определяется их число, которое заносится в память. Запоминается также количество положительных пиков (максимумов амплитуды) в каждом повторении. Кроме того, в память записываются значения интервалов между первыми четырьмя парами положительных пиков. Все перечисленные показатели приведены на рис. 30.9. Процедура анализа построена так, что в расчетах участвуют только данные, относящиеся к положительным пикам. Можно было бы использовать не только положительные, но и отрица-

тельные пики (минимумы амплитуды), однако эксперименты показали, что необходимости в этом нет.

Обозначив общее число повторений (повт.), наблюдающихся в пределах образа, через R , запишем формулы, по которым рассчитываются первые четыре параметра:

$$r_1 = R / \text{число пиков } P_2 \text{ в } R \text{ повт.};$$

$$r_2 = R / \text{число пиков } P_3 \text{ в } R \text{ повт.};$$

$$r_3 = R / \text{число пиков } P_4 \text{ в } R \text{ повт.};$$

$$r_4 = \text{суммарное число пиков } (P) \text{ в } R \text{ повт.} / R.$$

Не исключено, что в некоторых из этих выражений знаменатель окажется равным нулю, ввиду чего в программе перед выполнением деления должны быть предусмотрены соответствующие проверки.

Во всех вычислениях участвуют только целые числа, операции над которыми обычно занимают меньше времени, чем над числами с плавающей запятой. Именно поэтому сделано так, чтобы значения всех параметров были не меньше 1. Последний из параметров дает среднее число пиков в каждом из затухающих колебательных процессов. Это число играет важную роль при определении типа гласного звука или образа. Если в последнем обнаруживается более 16 участков, содержащих затухающие колебания ($R > 16$), то этот образ с большой вероятностью соответствует гласному звуку.

Затухающие колебания — анализ временных интервалов. Вторые четыре параметра вычисляются, подобно первым, и в их формулах используются те же обозначения. Эти параметры определяют средние значения интервалов между каждой из первых трех пар пиков, а также по всему образу в целом:

$$r_5 = \text{сумма } \Delta t_1 \text{ в } R \text{ повт.} / \text{число пиков } P_2 \text{ в } R \text{ повт.};$$

$$r_6 = \text{сумма } \Delta t_2 \text{ в } R \text{ повт.} / \text{число пиков } P_3 \text{ в } R \text{ повт.};$$

$$r_7 = \text{сумма } \Delta t_3 \text{ в } R \text{ повт.} / \text{число пиков } P_4 \text{ в } R \text{ повт.};$$

$$r_8 = \text{сумма всех } \Delta t \text{ в } R \text{ повт.} / R.$$

И вновь перед выполнением деления в программе должна производиться проверка на равенство знаменателя нулю. Отметим также, что отношение r_8 представляет среднюю продолжительность затухающих колебаний в данном образе.

Для того чтобы продемонстрировать конкретные результаты вычисления параметров с r_1 по r_8 , возьмем в качестве примера изображенный на рис. 30.9 фрагмент сигнала, в котором трижды появляются затухающие колебания, т. е. $R = 3$. Числа пиков типа P_1 , P_2 , P_3 , P_4 равны соответственно 3, 3, 3 и 2. Значения

временных интервалов приведены на рисунке. Подстановка этих чисел в формулы для $\gamma_1 - \gamma_8$ дает следующие величины: 3/3, 3/3, 3/1, 13/3, 23/3, 28/2, 20/2, 104/3.

Пиковые значения — анализ амплитуд. Следующие четыре параметра ($\gamma_9 - \gamma_{12}$) характеризуют свойства пиковых значений речевого сигнала. В данном случае учитываются как положительные (максимумы), так и отрицательные пики (минимумы). При этом параметры вычисляются как отношения числа пиков определенного типа к общему числу пиков P во всех R повторениях. Отметим, что в формуле для γ_{10} под «равенством» пиков подразумевается, что двоичные представления их амплитуд совпадают с точностью не менее 6 разрядов из 8; иными словами, соседние пики должны отличаться не более чем в 4 раза.

Параметры $\gamma_9 - \gamma_{12}$ описываются следующими соотношениями:

$\gamma_9 = P / \text{число почти нулевых соседних пиков во всем образе};$

$\gamma_{10} = P / \text{число равных соседних положительных пиков во всем образе};$

$\gamma_{11} = P / \text{число соседних монотонно изменяющихся пиков (положительных или отрицательных) во всем образе};$

$\gamma_{12} = P / \text{число соседних пиков с противоположными наклонами во всем образе}.$

Последняя формула нуждается в разъяснении. При вычислении ее знаменателя анализируются тройки соседних пиков. Если, например, $P_1 > P_2 < P_3$, то в первом промежутке ($P_1 - P_2$) наклон отрицательный (сигнал убывает), во втором ($P_2 - P_3$) — положительный (сигнал нарастает). Подобная ситуация и имела в виду, когда говорилось о противоположных наклонах. Аналогичным способом проверяются точки P_4 , P_5 и т. д.

Пиковые значения — анализ временных интервалов. Параметры, получаемые путем анализа временных интервалов, схожи с четырьмя предыдущими. Поскольку число пиков и интервалов между ними отличаются всего на 1, в вычислениях используется величина P . Параметры определяются по следующим формулам, в которых термины «равенство» и «противоположные наклоны» имеют тот же смысл, что и прежде:

$\gamma_{13} = P / \text{число соседних } \Delta t, \text{ меньших } 280 \text{ мкс, во всем образе};$

$\gamma_{14} = P / \text{число соседних } \Delta t, \text{ больших или равных } 280 \text{ мкс, во всем образе};$

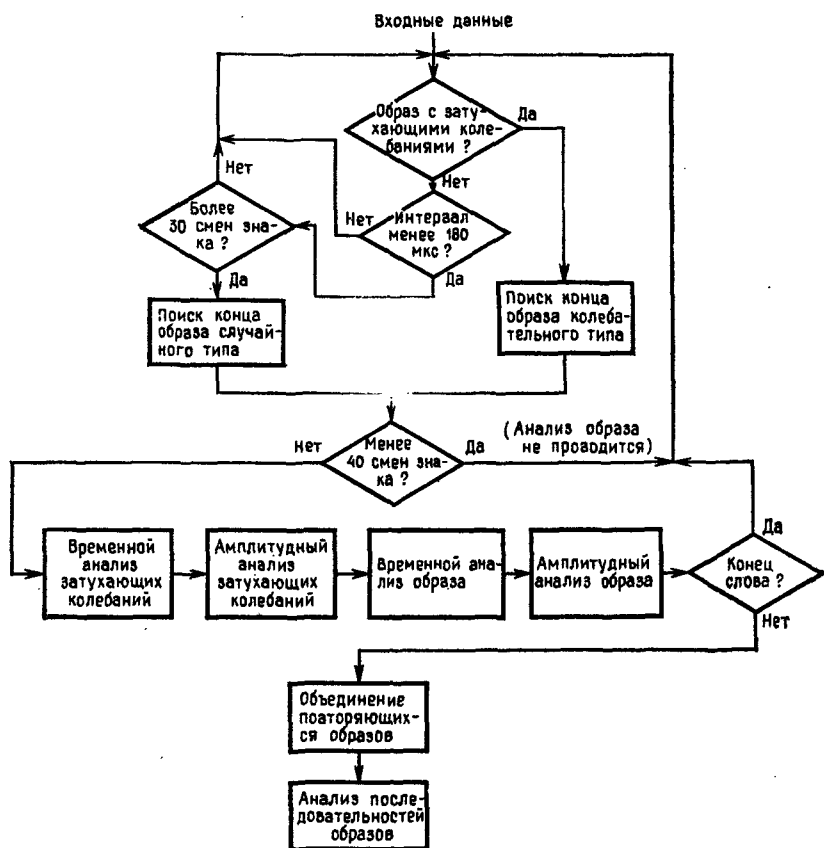


Рис. 30.10. Схема процедуры классификации образов.

$\tau_{15} = R/\text{число } \Delta t \text{ между соседними монотонно изменяющимися пиками (положительными или отрицательными) во всем образе;}$

$\tau_{16} = R/\text{число соседних } \Delta t \text{ с противоположными наклонами во всем образе.}$

Процедура вычисления параметров. Последние два параметра представляют число повторений в пределах образа (ранее оно было обозначено через R) и интервал между пиками, усредненный по всему образу. Последний определяется по формуле $\tau_{17} = \text{сумма } \Delta t \text{ во всем образе}/R$.

Схема описанной вычислительной процедуры изображена на рис. 30.10. Алгоритмы определения границ слова и сжатия ин-

формации (селекции) рассматривались ранее, и их подробные схемы можно увидеть на рис. 30.6. Микропроцессорная программа, реализующая процедуру, выполняется очень быстро. Она занимает всего около 300 строк на языке ассемблера и требует для своего хранения примерно 600 байт памяти.

Анализ образов. Специальные процедуры, с помощью которых должны обрабатываться описанные параметры, пока находятся в стадии разработки, поэтому еще нет экспериментальных результатов, опираясь на которые можно было бы судить о том, достаточно ли этих 18 параметров для достоверной классификации звуков речи. Возможно, практическое исследование алгоритмов анализа образов и распознавания образов покажет, что набор параметров неполон или избыточен, либо некоторые из параметров должны определяться по-другому. Тем не менее некоторые данные, полученные в ходе проведения работ, дают основания надеяться, что предложенный набор несет достаточную информацию. Исходя из результатов ручных расчетов и сравнения параметров, описывающих слова различного звучания, можно сделать ряд выводов, имеющих важное значение для дальнейшей разработки процедур анализа.

Описание любого образа включает обширный набор параметров и при определении типа образа все они должны приниматься в расчет. Поскольку образ характеризуется множеством взаимосвязанных признаков, его следует анализировать как единое целое.

Главной особенностью глухих согласных (например, $|t|$ в слове two) является относительная малость параметров γ_{13} и γ_{14} . В словах, подобных *ope* и *four*, соответствующие образы имеют близкие параметры $\gamma_9 - \gamma_{16}$, а также схожие амплитудные характеристики колебательных процессов ($\gamma_1 - \gamma_4$), но существенно отличаются во временном описании колебаний ($\gamma_5 - \gamma_8$). В звуках, произносимых без придыхания, встречается значительное число случайно чередующихся переключений наклона сигнала. Кроме того, для них характерно малое среднее значение интервала между пиками в пределах образа (параметры γ_{13} и γ_{17}).

Эти наблюдения можно проиллюстрировать на примере реальных параметров, полученных при анализе слова *ship*. Значения, выданные системой, приведены в табл. 30.1. В первом образе содержится большое число пиков с противоположными наклонами (γ_{12}). Средний интервал изменения наклона велик (γ_{17}), и вообще данный образ отличается короткими промежутками между пиками (γ_{13}). Высокая повторяемость затухающих колебаний (R и γ_{15}) говорит в пользу случайного происхождения сигнала, поскольку в случайном образе могут присутствовать процессы любых типов, в том числе и колебательные.

Таблица 30.1. Параметры слова ship

Затухающие колебания											Пиковые значения									
Амплитуды						Временные интервалы					Амплитуды				Временные интервалы					
τ_1 τ_2 τ_3 τ_4 R						τ_6	τ_7	τ_8	τ_9	τ_{10}	τ_{11}	τ_{12}	τ_{13}	τ_{14}	τ_{15}	τ_{16}	τ_{17}			
Образ	1	1	1	1	3	109	4	5	6	38	15	2	8	2	2	8	3	16	23	
	2	1	1	2	3	5	21	32	44	80	42	42	2	2	13	40	2	2	37	
	3	1	1	1	2	12	8	27	34	128	0	0	1	2	50	50	2	2	39	
	4	1 ¹⁾	1	1	4	3	7	32	16	64	0	12	2	3	...	10	2	2	48	
¹⁾ Обозначает паузу внутри фоны.																				

¹⁾ Обозначает паузу внутри фонемы.

Все эти соображения дают основание полагать, что образ соответствует глухому согласному звуку [s], [t] или [f]. Уточнить решение помогают параметры τ_5 — τ_8 , из анализа которых следует, что наиболее вероятным является звук [s]. Образы 2 и 3 представляют среднюю часть слова, причем последний отмечен флажком, который указывает, что далее следует взрывная согласная (флажок устанавливается в процессоре ввода речевой информации).

Можно утверждать, что для правильной работы как рассматриваемой, так и любой другой системы распознавания речи, принципиальное значение имеет высокая достоверность идентификации гласных звуков. Существует множество схожих по написанию слов, отличающихся друг от друга лишь одной гласной, например hit, hat, hot. Как следует из табл. 30.2, половина фонем представляет гласные или близкие к ним звуки (т. е. сами гласные, полугласные и дифтонги). И хотя звуки этих типов несколько отличаются числом повторений колебательных процессов, для более точной идентификации необходимы методы, позволяющие исследовать частотный состав образов.

Многие подходы к проблеме распознавания речи опираются именно на частотный анализ звуковых сигналов. Он основан на непосредственном (с помощью набора фильтров) или косвенном (за счет использования быстрого преобразования Фурье) оценивании энергетического спектра сигнала. Как правило, в гласных звуках преобладают три (иногда больше) полосы частот, именуемых формантными, в которых сосредоточена значительная часть энергии сигнала. Наиболее выраженные частоты в этих полосах для различных гласных звуков указаны в табл. 30.3. Все приведенные цифры получены путем усреднения по результатам опытов, в которых разными голосами произносились различные последовательности слов.

Таблица 30.2. Мнемонические обозначения звуков в американизированном варианте английского языка

Гласные		Полугласные		Дифтонги	
Обозначение	Пример	Обозначение	Пример	Обозначение	Пример
IV	beet	W	wag	eI	bay
I	bit	L	lamb	oU	boat
E	bet	R	rod	aI	buy
AD	bat	Y	yes	aU	how
UH	but			oI	boy
A	hot			iU	few
OW	bough	Произношение этих звуков существенно зависит от позиции фонемы в слове, а также от обрамляющих гласных			
U	foot				
OO	boot				
ER	bird				

Согласные					
Сонанты		Взрывные		Гортанный глухой	
Обозначение	Пример	Обозначение	Пример	Обозначение	Пример
M	miss	B	big	H	hand
N	knot	D	dig		
NG	sing	G	give		
		P	pin		
		T	tin		
		K	cat		

Фрикативные		Аффрикаты	
Обозначение	Пример	Обозначение	Пример
V	van	DZH	gem
TH	then	TSH	chin
Z	zip		
ZH	rouge		
E	fan		
THE	thin		
S	sod		
SH	shove		

Для разработанного варианта системы переход к описанию сигналов в частотной области был сочтен нецелесообразным, ввиду чего все расчеты выполняются во временной области. Следует, однако, полагать, что в параметрах, связанных с временными интервалами, содержится также определенная информация о частотных характеристиках сигналов. И хотя строгие

Таблица 30.3. Типовые значения формантных частот для гласных звуков

Обозначение звука	Пример	Формантные частоты		
		F ₁	F ₂	F ₃
IV	beet	270	2290	3010
I	bit	390	1990	2550
E	bet	530	1840	2480
AE	bat	660	1720	2410
UH	but	520	1190	2390
A	hot	730	1090	2440
OW	bought	570	840	2410
U	foot	440	1020	2240
OO	boot	300	860	2240
ER	bird	490	1350	1690

математические зависимости пока не выведены, весьма возможно, что усредненные показатели, получаемые в результате анализа различных интервалов, имеют непосредственное отношение к формантным частотам. Это подтверждается тем, например, что продолжительность первого цикла колебаний, усредненная по всем повторениям, соответствует третьей форманте, а параметр τ_8 — первой.

Последовательности фонем. Обсуждаемый подход предполагает существование определенной взаимосвязи между образами и фонемами. Однако описанные выше процедуры не способны однозначно выбрать конкретную фонему, соответствующую данному образу. Невозможность абсолютно полной идентификации фонем объясняется и тем, что образы искажаются из-за частичного наложения соседних фонем, и тем, что методы анализа образов пока далеки от совершенства.

Тем не менее можно полагать, что эти процедуры позволяют существенно сузить круг потенциальных фонем, отвечающих данному образу. С их помощью будут формироваться цепочки фонем-кандидатов, в той или иной степени аппроксимирующие истинную последовательность фонем, содержащихся в слове. Далее эти цепочки совместно с вычисленными параметрами предполагается использовать для организации процедуры распознавания слова путем просмотра специального словаря, который включал бы не только эталонные цепочки фонем, но и возможные отклонения от этих эталонов.

В качестве конкретного примера можно взять слово *six* (читается как *|siks|*), которое легко расчленяется на четыре фонемы: звук случайного типа, гласный звук, взрывной, согласный (на это указывает пауза в начале фонемы), и вновь звук случайного типа. Рассмотренная ранее процедура идентифика-

ции может дать достаточно точную информацию о присутствии в слове гласного звука. Точно так же анализ звуков случайного типа позволит выбрать наиболее вероятный фриктивный согласный, отделив, в частности, [s] от [z].

Весьма желательно, чтобы алгоритм, применяемый для идентификации последовательностей фонем, обладал некоторыми «интеллектуальными» свойствами. Дело в том, что многие образы лишены смысла, поскольку они отражают лишь переходные процессы, в то время как другие являются простыми повторениями уже встречавшихся ранее образов. Следует также учитывать, что точное совпадение параметров с эталонными из-за индивидуальных различий в произношении крайне маловероятно. Возможно лишь сравнение ключевых параметров, описывающих образы, и поиск фонем, максимально близких к эталонным в смысле избранного критерия сходства.

30.10. МЕТОДИКА ИДЕНТИФИКАЦИИ СЛОВ

На завершающем этапе процесса распознавания речи отобранные последовательности фонем должны сопоставляться с реальными словами. Для этой цели необходим специальный словарь, содержащий эталонные последовательности, а также другие данные, используемые при идентификации. Должна быть разработана процедура поиска, с помощью которой из этого словаря можно было бы извлечь слово, обладающее заданной степенью сходства с предъявленной последовательностью.

Для решения указанной задачи целесообразно применять не 8-разрядные, а более мощные 16- или 32-разрядные микропроцессоры. Они имеют расширенные наборы команд, более гибкие системы адресации и работают со словами большей длины. Все это вместе взятое позволяет сократить длительность вычислений и снизить требования к емкости оперативной памяти.

Объем памяти, которая должна быть отведена для размещения словаря, в первую очередь зависит от применяемого алгоритма поиска и от сложности принятого критерия сходства, определяющего степень «интеллектуальности» системы. В простейших системах, предназначенных для распознавания цифр, словарь уместается в нескольких сотнях байтов. Что же касается сложных систем, способных воспринимать предложения или фразы, то для них необходимы колоссальные многомегабайтные словари, обеспечивающие возможность проведения синтаксического разбора, т. е. разделения предложений на существительные, глаголы и другие части речи.

Существует немало хорошо себя зарекомендовавших методов поиска. В некоторых случаях целесообразно использовать их совместно, чередуя в рамках единой процедуры. Ниже кратко

описываются три метода, которые в первую очередь можно рекомендовать для применения в системах распознавания.

Половинное деление. Метод половинного деления оказывается весьма эффективным, если количество анализируемых элементов сравнительно невелико, что типично, например, для задачи распознавания произносимых вслух чисел. Таблицу, объединяющую эти элементы, можно разместить в оперативной памяти. Поиск начинается с того, что таблица делится на две половины, после чего выясняется, в какой из них может содержаться обнаруженная совокупность фонем. Выбранная половина затем вновь делится пополам и процедура повторяется вновь и вновь, пока не будет найдено искомое слово. При таком способе поиска таблица из 1024 элементов обрабатывается всего за 10 шагов.

Индексирование. Метод индексирования целесообразно применять для просмотра словарей большого объема. Составляется таблица (индекс), по которой находится адрес группы слов, обладающих некоторым общим свойством. Такая таблица может, например, объединять всевозможные фонемы, соответствующие тем звукам, с которых начинаются слова. Если первый образ, содержащийся в речевом сигнале, классифицирован как звук [s], то в таблице отыскивается адрес списка слов, начинающихся с этого звука. Данный список должен содержать типовые совокупности фонем, из которых состоят эти слова, а также другие параметры, играющие роль ключей. Каждый элемент списка может еще быть дополнен символами, образующими запись слова на английском языке. Эту запись затем можно либо просто выводить на печать, либо использовать для выполнения каких-то дальнейших действий.

Хеширование параметров. Метод хеширования основан на том, что вся совокупность ключевых параметров (в данном случае она представляет последовательность фонем) преобразуется, согласно определенному правилу, в адрес искомого элемента. Ключевые параметры могут включать порядковые номера эталонных образов, позиции гласных в слове, звуки, с которых начинаются слова, и другую информацию. Алгоритмы хеширования являются весьма эффективным средством преобразования больших наборов параметров, занимающих, скажем, 40—50 байт, в стандартные 16- или 24-разрядные адреса.

30.11. АППАРАТНАЯ РЕАЛИЗАЦИЯ

Можно предложить много различных схем аппаратной реализации системы распознавания речи. Одна из возможных конфигураций представлена на рис. 30.1. Теперь, когда мы обсудили основные задачи, возникающие в процессе распознавания, а

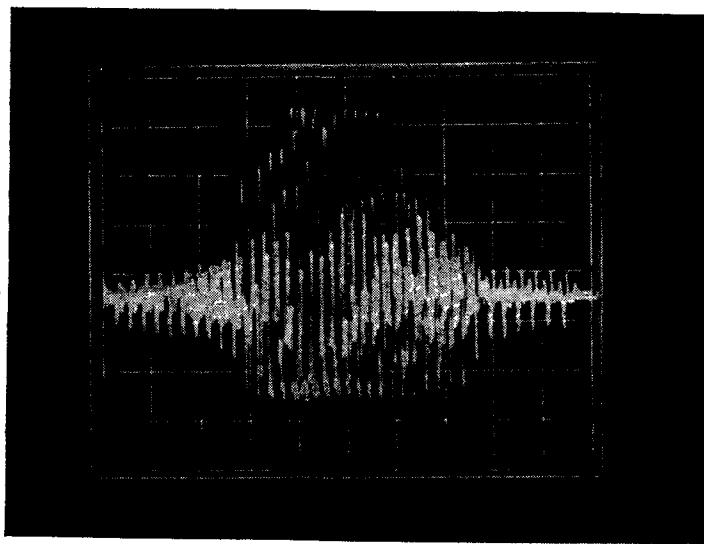


Рис. 30.11. Звуковые образы в слове опе.

также возможные методы их решения, можно более подробно остановиться на предложенном варианте построения аппаратной части. В разработанной системе параметры, определяющие отдельное слово, формируются в течение примерно 1,5 с. Однако следует учитывать, что благодаря конвейерной структуре системы в ней одновременно может обрабатываться целая последовательность слов.

Первый процессор системы осуществляет первичную обработку входной информации, которая поступает с аналого-цифрового преобразователя, снабженного оперативной памятью (ОЗУ) объемом 128 байт и постоянной памятью (ПЗУ) объемом 512 байт. Преобразователь работает с частотой 50 Гц. Программа процессора обеспечивает выполнение таких операций, как линейное экстраполирование входных сигналов с использованием переменного опорного напряжения, определение границ слов и сжатие информации. В результате каждую секунду формируется примерно 500—1000 параметров, которые пересылаются для дальнейшей обработки в следующий процессор.

Процессор 2 имеет ОЗУ на 8К байт и ПЗУ той же емкости. Он реализует алгоритм выделения звуковых образов, описываемых 18 параметрами. Анализ текущего слова сопровождается приемом информации, относящейся к следующему. Второй процессор посылает в третий от 100 до 200 параметров в секунду.

Процессор 3 играет в системе роль ведущего. Он оснащен стандартным набором периферийных устройств, включающих дисплей с клавиатурой и накопитель на жестких или гибких дисках. Этот процессор, имеющий ОЗУ объемом в 64К байт, предназначен для реализации алгоритма поиска, построенного на основе двух методов — индексирования и хеширования. Кроме того, на этом же процессоре выполняются программы, обеспечивающие связь пользователя с системой.

30.12. ТЕКУЩЕЕ СОСТОЯНИЕ ПРОЕКТА

Как уже не раз отмечалось, предложенная система еще не завершена. В настоящее время решены все проблемы, связанные с первыми двумя процессорами, т. е. последовательность обработки реализована вплоть до вычисления параметров, описывающих звуковые образы. Ведется также работа над созданием управляющего программного обеспечения.

Поскольку система еще не доведена до конца, трудно говорить о точности, с которой она сможет распознавать слова человеческой речи. Вполне вероятно, что качество работы первого варианта системы окажется ниже ожидаемого. В этом случае потребуются новые исследования, которые подскажут, какие изменения следует ввести в процедуры, описанные в настоящей главе. Более того, возможно, что ограниченные возможности существующих микропроцессоров или принципиально неверное толкование результатов предварительных экспериментов, расчетов и эвристических выкладок вообще не позволят добиться поставленной цели.

Тем не менее обсуждаемая экспериментальная система полезна хотя бы тем, что она открывает новые перспективы применения микропроцессоров. Ведь несмотря на то что исследования в области распознавания речи ведутся уже свыше двенадцати лет, реальных широкодоступных универсальных систем, обладающих большим словарным запасом, так пока и не создано. Однако можно не сомневаться, что когда-нибудь к услугам проектировщиков будет разнообразный выбор микропроцессорных систем, которые сыграют важную роль в решении многих прикладных задач. Ну, а пока проблема распознавания речи остается предметом интересных и многообещающих исследований для энтузиастов в этой области.

В заключение я хотел бы выразить признательность моей жене Боните за консультации в вопросах, связанных с лингвистикой, а также д-ру Т. Дж. Харрисону за помощь в подготовке рукописи этой главы.

ЛИТЕРАТУРА

1. Dinneen F. P., An Introduction to General Linguistics, Holt, New York, 1967.
2. Dixon N. R., Martin T. B., Automatic Speech and Speaker Recognition, IEEE Press, New York, 1979.
3. Rabiner L. R., Schafer R. W., Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, N. J., 1978.
4. Harrison T. J., Handbook of Industrial Control Computers, Wiley-Interscience, New York, 1972.
5. Heilman A. W., Principles and Practices of Teaching Reading, Merrill, Columbus, Ohio, 1972.
6. Peterson G. E., Barney H. L., «Control Methods Used in a Study of the Vowels», J. Acoust. Soc. Am., Vol. 24, No. 2, March 1952, pp. 175—184.

СПИСОК ТЕРМИНОВ

Ада (Ada): разработанный по заказу министерства обороны США язык программирования, являющийся развитием Паскаля. Подобно Паскалю, он отличается высокой структурированностью и наличием механизмов, повышающих его мобильность.

Адрес (address): идентификатор, определяющий ячейку памяти или источник (приемник) данных.

Аккумулятор (accumulator): регистр, предназначенный для временного хранения результатов операций, выполняемых в арифметико-логическом устройстве (АЛУ) процессора.

Акустический преобразователь (acoustic coupler): устройство, преобразующее выдаваемые ЭВМ коды в электрические сигналы звуковой частоты для передачи их по радио или телефонным каналам.

Алгоритм (algorithm): четко определенная совокупность правил или процессов для решения задачи.

АЛУ (ALU): сокращенное обозначение арифметико-логического устройства.

Аналоговый (analog): в информатике и электронике тип сигналов или данных, изменения которых носят непрерывный характер, например температура, напряжение и т. д.

Аналого-цифровой преобразователь (analog to digital converter): устройство, преобразующее аналоговые данные в цифровые коды.

Аномальное окончание (ABEND): прекращение выполнения программы до ее нормального завершения, вызванное ошибочной ситуацией.

Аппаратное обеспечение (hardware): комплекс аппаратуры, входящий в состав вычислительной системы.

Арифметико-логическое устройство (arithmetic and logic unit): часть центрального процессора, предназначенная для выполнения арифметических и логических операций. Их результаты заносятся в аккумулятор.

Архитектура (architecture): внутренняя структура и организация микропроцессора или другой интегральной схемы.

Ассемблер (Assembler): программа, предназначенная для перевода команд языка ассемблера в машинные команды, выполняемые процессором.

АЦП (ADC): сокращенное обозначение аналого-цифрового преобразователя.

База данных (data base): информационный файл большого объема, обрабатываемый с помощью ЭВМ.

Байт (byte): 8-битовое «слово».

Библиотека (library): совокупность программ, образующих единый файл.

БИС (LSI): сокращенное обозначение большой интегральной схемы, содержащей тысячи логических компонентов в одном чипе.

Бит (bit): двоичная цифра (обычно 0 или 1) — наименьшая единица информации в ЭВМ.

Бод (baud): единица скорости передачи дискретной информации, обычно принимаемая равной одному биту в секунду.

Буфер (buffer): цепь, используемая для изоляции или сопряжения двух или нескольких различных элементов аппаратуры.

Буфер данных (data buffer): регистр или небольшое запоминающее устройство, предназначенное для временного хранения данных при обмене информацией между ЦП и внешними устройствами, обладающими меньшим быстродействием.

Бейсик (BASIC): многоцелевой язык символических команд для начинающих.

Внешнее устройство (peripheral): устройство памяти или ввода-вывода, подключаемое к ЦП: клавиатура, видеомонитор, диск, печатающее устройство и т. д.

Выполнение (execution): реализация действия, предписанного программой или оператором ЭВМ.

ГАП (CAM): сокращенное обозначение гибкого автоматизированного производства.

Генератор (clock): цепь, вырабатывающая периодические синхросигналы, используемые в различных устройствах ЭВМ.

Гибкий диск (floppy disk): покрытый магнитным материалом пластиковый диск, используемый в микроЭВМ в качестве внешнего носителя информации большой емкости.

Двоичный (binary): тип сигналов или условий, способных принимать одно из двух возможных состояний или значений.

Диагностический (diagnostic): тип программы или сообщения, предназначенных для выявления и классификации ошибок в программе или отказов в вычислительной системе.

Динамический (dynamic): тип памяти, в которой данные представлены зарядами на конденсаторах. Из-за утечки зарядов данные могут искажаться, ввиду чего подобная память нуждается в периодической «регенерации».

Диск (disk): покрытый магнитным материалом диск, используемый в вычислительных системах в качестве носителя информации.

Дискетта (diskette): название дисков небольшого размера, чаще всего диаметром 133 и 203 мм.

Жесткий диск (hard disk): покрытый магнитным материалом металлический диск, используемый в качестве носителя информации большой емкости.

Загрузчик (loader): программа, осуществляющая поиск скомпонованной программы во внешней памяти и загрузку ее в оперативную память для последующего выполнения.

Интерпретатор (interpreter): программа, последовательно транслирующая и выполняющая предложения языка высокого уровня.

Интерфейс EIA (EIA interface): интерфейс для терминалов и модемов, отвечающий стандартам Ассоциации предприятий электронной промышленности (США).

Килобайт (kilobyte): единица объема памяти, равная 1024 байтам (сокращенно обозначается К).

КМОП (CMOS): сокращенное обозначение комплементарных структур металл — окисел — полупроводник. Благодаря малому энергопотреблению и низкой чувствительности к изменению напряжения питания микросхемы на КМОП-структурах получили широкое распространение в портативных микроЭВМ, в том числе с батарейным питанием.

Код операции (op code): записанный в двоичной, восьмеричной или шестнадцатеричной системе код команды процессора.

Команда (instruction): мнемоническая запись или код, задающие тип операции, подлежащей выполнению в ЦП, а также участвующие в ней данные.

Компилятор (compiler): программа, предназначенная для перевода операторов языка высокого уровня в машинные команды, выполняемые процессором.

Компоновщик (linker): программа, осуществляющая формирование готовой к выполнению программы из отдельных блоков, представленных в машинном коде.

Контрольный бит (parity): дополнительный бит, используемый для контроля правильности записи элемента данных.

Кросс-ассемблер (cross assembler): программа, предназначенная для перевода команд языка ассемблера процессора одного типа в машинные команды, выполняемые процессором другого типа.

Макроассемблер (macro-assembler): ассемблер, способный обрабатывать макрокоманды.

Макрокоманда (macro-instruction): команда языка ассемблера, эквивалентная определенной последовательности команд процессора.

Массив (array): упорядоченный набор данных или других элементов.

Массовая память (mass storage): совокупность внешних устройств, предназначенных для хранения и выборки информации большого объема.

Машинный код (machine code): группа данных, интерпретируемых как команды, выполняемые процессором.

Микропроцессор (microprocessor): центральный процессор микроЭВМ.

МикроЭВМ (microcomputer): вычислительная система, реализованная на одной или нескольких БИС.

Мобильность (portability): показатель, характеризующий возможность переноса программного обеспечения, созданного для определенной вычислительной системы, на систему другого типа.

Модем (modem): сокращенное обозначение модулятора-демодулятора. С помощью этого устройства цифровые коды, выдаваемые ЭВМ, преобразуются в электрические сигналы звуковой частоты, которые передаются по линиям связи и затем вновь переводятся в цифровые коды.

Монитор (monitor): программа, реализующая основные служебные функции управления и контроля работы вычислительной системы.

Объектная программа (object program): машинный код, формируемый компилятором или интерпретатором в результате обработки программы на исходном языке.

ОЗУ (RAM): сокращенное обозначение оперативного запоминающего устройства, допускающего как считывание, так и запись информации.

Оператор (statement): законченное содержательное выражение на языке высокого уровня.

Операционная система (operation system): совокупность программ, управляющих функционированием всех компонентов вычислительной системы (обрабатывающих программ, средств ввода-вывода и т. д.).

Опрос (polling): метод периодического обращения к внешним устройствам с целью определения их текущего состояния.

Отладка (debugging): процесс выявления и коррекции ошибок в программе ЭВМ.

Память (memory): часть вычислительной системы, предназначенная для хранения данных.

Параллельный (parallel): метод обмена данными, при котором все биты, образующие слово, пересылаются одновременно.

Параметр (parameter): переменная, которой при каждом выполнении подпрограммы присваивается определенное значение; по завершении подпрограммы данное значение может передаваться другим переменным основной программы.

Передача управления (branch): команда, после которой выполняется не следующая по порядку команда, а расположенная в другой точке программы.

Переход (jump): синоним термина «передача управления».

Подпрограмма (subroutine): вызываемая основной программой последовательность операторов, оформленная в виде отдельной программы.

Порт (port): электронный узел, через который осуществляется связь между ЦП и различными устройствами ввода-вывода.

Последовательный (serial): метод обмена данными, при котором биты, образующие слово, пересылаются поочередно, один за другим.

ППЗУ (PROM): сокращенное обозначение программируемого постоянного запоминающего устройства, разновидности ПЗУ, допускающего перепрограммирование в процессе эксплуатации.

Прерывание (interrupt): приостановка нормального процесса выполнения программы, вызванная содержащейся в ней командой или поступившим извне сигналом.

Прикладная программа (application program): программа, предназначенная для решения на ЭВМ определенной прикладной задачи.

Программа (program): совокупность команд, задающих последовательность действий ЦП, целью которых является получение требуемого результата.

Программно-аппаратное обеспечение (firmware): набор программ, хранящихся в постоянном запоминающем устройстве (ПЗУ).

Программное обеспечение (software): комплекс системных программ и сопровождающей документации.

Прослеживание, трассировка (trace): метод отладки, при котором после выполнения каждой команды программа приостанавливается и выводится содержимое регистров.

Процедура загрузки (bootstrap): программа, содержащая команды, необходимые для загрузки операционной системы ЭВМ и подготовки ее к выполнению прикладных программ.

Процессор (processor): общее название части аппаратных средств вычислительной системы, реализующей основные операции обработки данных.

Прямой доступ к памяти (direct memory access): механизм доступа, посредством которого внешнее устройство может непосредственно, минуя ЦП, обращаться к оперативной памяти.

Пульт (console): устройство, используемое для связи оператора с ЭВМ.

Регистр (register): ячейка памяти или специальное устройство емкостью в одно слово, используемое для временного хранения данных в ходе выполнения программы.

Редактор (editor): программа редактирования текстовой информации, позволяющая запоминать тексты в файлах и вносить в них требуемые изменения.

РПГ (RPG): сокращенное обозначение генератора программ отчетов, языка программирования и обрабатывающей программы для генерации объектных программ, предназначенных для формирования, сопровождения и выдачи отчетов при решении экономических задач.

САПР (CAD): сокращенное обозначение системы автоматизированного проектирования.

Система команд (instruction set): полный набор команд, реализованных в процессоре.

Слово (word): единица информации стандартной длины, обычно равной 4, 8, 16 или 32 бит.

Служебная программа, утилита (utility): программа, выполняющая определенную служебную функцию, например сверку текстов, перезапись файлов и т. д.

СППЗУ (EPROM): сокращенное обозначение стираемого программируемого запоминающего устройства, разновидности ППЗУ, в котором информация может стираться под действием электрических сигналов или облучения ультрафиолетовым светом.

Стек (stk): группа регистров или ячеек памяти, используемых для хранения адресов или данных.

Строка (string): последовательность алфавитно-цифровых символов.

Сумматор (adder): электронная цепь, на выходе которой формируется код, представляющий сумму двух или большего числа входных кодов.

Схема программы (flowchart): схема, отображающая последовательность операций, выполняемых программой, предназначенной для решения определенной задачи

Считывание (read): процесс выборки данных из запоминающего устройства.

Текстовый редактор (text editor): программа, с помощью которой символьная информация может вводиться в память ЭВМ, редактироваться и запоминаться в файле.

Терминал (terminal): устройство, позволяющее вводить в ЭВМ программы, передавать их на выполнение и выводить получаемые результаты.

Точка приостанова (breakpoint): точка, в которой выполнение программы может быть прервано и осуществлена проверка некоторых условий.

Указатель (pointer): регистр, предназначенный для хранения адреса данных или команды в оперативной памяти.

Указатель данных (data pointer): регистр, содержащий адрес данных, подлежащих обработке.

Файл (file): совокупность связанных данных, хранящихся во внешней памяти.

Флаг (flag): бит, используемый в качестве индикатора состояния устройства или результата операции.

Центральный процессор (central processing unit): часть процессора, предназначенная для выполнения команд и управления работой АЛУ, аккумулятора и других регистров.

Цифровой (digital): тип сигнала или данных, изменения которых в отличие от аналоговых носят ступенчатый характер.

ЦП (CPU), сокращенное обозначение центрального процессора.

Чип, кристалл (chip): кремниевая пластинка, на которой формируется интегральная схема.

Шина (bus): совокупность проводников, предназначенных для перемещения данных между различными устройствами ЭВМ.

ЭЛТ (CRT): сокращенное обозначение электронно-лучевой трубки — прибора, составляющего основу многих устройств визуализации, применяемых в вычислительной технике.

Эмулятор (emulator): комплекс аппаратных и программных средств, обеспечивающий выполнение команд микропроцессора одного типа с помощью ЭВМ, выполненной на микропроцессоре другого типа.

Энергонезависимая память (non-volatile memory): память обеспечивающая сохранность информации при отключении электропитания.

ЭПЗУ (EAROM): сокращенное обозначение электрически перепрограммируемого постоянного запоминающего устройства, информация в которое может заноситься с помощью электронных сигналов, но не стирается при отключении электропитания.

ЭСЛ (ECL): сокращенное обозначение логических схем с эмиттерными связями, отличающихся малым временем срабатывания. Микропроцессоры, выполненные по этой технологии, применяются в быстродействующих ЭВМ.

ЭСПЗУ (EEPROM): сокращенное обозначение электронно-стираемого программируемого постоянного запоминающего устройства, разновидности ПЗУ, в котором информация может стираться под действием электронных сигналов.

Эталонная программа (benchmark): программа, которую можно выполнять на различных ЭВМ с целью сравнения их быстродействия и производительности.

Язык ассемблера (assembly language): язык, содержащий символические команды, которые программа-ассемблер переводит в команды процессора.

Язык высокого уровня (high level language): язык для записи программ ЭВМ, содержащий элементы естественного языка.

ANSI: сокращенное обозначение Американского национального института стандартов.

ASCII: сокращенное обозначение Американского национального стандартного кода для обмена информацией — системы кодирования алфавитно-цифровых и специальных символов с помощью 7-битовых «слов»

EBCDIC: сокращенное обозначение расширенного двоично-кодированного десятичного кода для обмена информацией — системы кодирования символов с помощью 8-битовых «слов», используемой в вычислительных системах и связанной с ними аппаратуре передачи данных.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- Авиатренажер** 9
Аккумулятор 123, 124, 249
Акустический тракт 346, 347
Алгоритм 14
— *Брезенхэма* 31
— вычерчивания отрезков 25—33
— генерации окружностей 33—35
Амплитудно-частотная сепарация 368, 371, 372
Аналого-цифровой преобразователь (ЦАП) 300—303
Ассемблер 203—246, 371
- База**
— данных 62
— правил (знаний) 62
Буфер образа 12
- Выделение слов** 371, 372
- Генератор**
— векторов 41, 111
— звуковой 351, 356
— напряжения 296—300
— окружностей 33
— символов 59
— синхронимпульсов 325—327
Гласные звуки 373, 374
Голосовой аппарат 346, 347
Графическая программа 109—118
Графический планшет 17
— терминал 9—12
— — интерактивный 11, 12
- Графопостроитель** 7, 99
— векторный 100
— инкрементный 100
— многоперьевой 100
— перьевой 99, 100
— печатающий 66, 103
— планшетный 103
— растровый 101
— рулонный 103
— электростатистический 102, 105
- Двончно-десятичный код** 154
Дешифратор команд 125, 323
Диалог 8
Дисплей векторный 35—37
— — с регенерацией изображения 54—59
— графический 9—12
— растровый 22, 24—35
— с запоминающей трубкой 51—54
— система команд 52—54
— со сканирующим лазером 49—51
Дисплейный контроллер 12, 37, 55—59
— процессор 56
— файл 55
Дифтонг 349
- Запись** 257
Запоминающая сетка 46
Запоминающее устройство оперативное (ОЗУ) 272
— — постоянное (ПЗУ) 272
Звуковой образ 374
— — идентификация 375—377, 383—387

- — параметры 378—382
- — типы 377, 378
- звуковой ответчик 342

Идентификация слов 387, 388

Изображение 14—17

- графическое 66
- последовательный вывод 22
- преобразование 16, 17
- растровое 22
- регенерация 39
- твердого тела 20, 21
- тонирование 20

Интегратор 41

Интенсивность свечения 39

Интерпретатор правил 62

Интерпретация языка 61

Интерфейс 248

- пользователя 19, 20

- с главной памятью 272—282

Искусственный интеллект 60—62

Кадрирование 16

Канал 308

Катод 39

Квантование 14, 360

- частота 364, 365

Коллектор 46

Команда арифметическая 217—232

- возврата из подпрограммы 57
- вывода текста 59
- обработки данных 131, 134, 135—138
- пересылки данных 131, 132, 232—247
- перехода к подпрограмме 57
- перехода с вталкиванием 57
- перехода с выталкиванием 58
- прерывания 128
- управления 131, 135, 139
- — безусловная 135
- — условная 135

Компоновщик 345, 356

Конвейерная система 361, 362

Линия выборки модуля 273

- прерывания 128
- разрешения 278
- управляющая 279
- чтения/записи 278

Логическое устройство управления (контроллер) 188—199, 215—217

Люминофор 38, 41, 42

Матрица 22

Машинная графика 7—14

- — интерактивная 7, 12—15
- — неинтерактивная (пассивная) 7

Машинные игры 7, 345

Метод итераций 24

- последовательных приближений 303

- превращений 22, 24, 25

Микропроцессор (МП) 119

- компоненты 122—130
- — арифметико-логическое устройство (АЛУ) 122—124
- — ввод-вывод 129, 130, 248—253
- — внешняя память 130, 131
- — регистры 124—126, 128
- — устройство управления 127, 128
- программное обеспечение 131—139
- секционированный 316—319
- система команд 131
- способы адресации 133, 134

МикроЭВМ 120, 143

- аппаратное обеспечение 146

- описание программ 147—152

- организация на базе МП 8080/8085 157—188

- программное обеспечение 131—139

- типы данных 152—155

- хранение данных 155, 156

Морфема 373

Мультиплексирование 303

Операционная система 328

- — для микрокомпьютеров 329
- — Apple 331
- — CP/M 333—337
- — MS-DOS 338, 339
- — OASIS 337, 338
- — UNIX 332, 340
- — XENIX 340, 341
- — компоненты 328, 329
- — — управление ресурсами 328
- — — человеко-машинный интерфейс 329
- — — ядро 328

Пакет графических подпрограмм 18, 111—118

Память, время выборки 278

- главная 272
- динамическая 281—283
- постоянная 272
- пространство 274, 275

— со считыванием и записью 272
 Передача данных 232, 308
 — — последовательная 308
 — — — асинхронная 311—316
 — — — синхронная 309—311
 Персональный компьютер 64, 65
 Печатающая головка 72—76
 Печатающее устройство 67
 — — безударное 67, 70—82
 — — лазерное 81
 — — посимвольное 69, 85—89
 — — построчное 69, 76—79, 79—82, 90
 — — струйное 94—96
 — — термическое 91
 — — точно-матричное 82—85
 — — ударное 67, 76—79, 85—94
 — — электростатическое 94
 — — электрочувствительное 91—93
 Пиксель 13
 Плазменная панель 47—49
 Плотность линии 26, 27
 Подвод бумаги 96—98
 Подпрограмма 200—206
 — аргументы 204
 — организация данных 204, 205
 — стек 201—203
 Порт ввода-вывода 129, 248—253
 — двунаправленный 129
 — однопорядочный 130
 Прерывание 128, 206—213
 — векторная система 262—269
 — инициализация 207
 — обработка 206—211
 — программы 260
 — запрос на 261
 — с программным опросом 261, 262
 — сигнал 206
 Проблема скрытых поверхностей 20
 Протокол взаимодействия 254
 Проявляющие электроны 47
 Прямой доступ к памяти 55, 284—289
 Пучок электронов 38
 — записывающий 46

Разрешающая способность 23
 Распознавание речи 358
 Реакция 15
 — скорость 16
 Регенерация изображения 39
 Регулируемые параметры 352—356
 Режим передачи последовательный 22
 — — — асинхронный 52
 Ресивер 315

Робот 60, 65
 Робототехника 63—65

Световое перо 17
 Сигнал прерывания 206
 — синхронизации 253
 — стробирующий 252
 Символ, оттиск 71
 — точно-матричный 69
 — целостный 69
 Синхронизатор музыки и шумовых
 эффектов 342, 344, 345
 — речи 342, 345—353
 Согласные звуки 373
 — — аффрикаты 385
 — — гортанный глухой 385
 — — носовые (сонанты) 352
 — — фрикативные 350
 Состояние 24
 Стекло 57
 — вершина 57
 — указатель 128
 Схема регенерации 283, 284

Твердая копия 66
 Теневая маска 43
 Трансмиссия 315

Управляющая сетка 39
 Управляющие байты 52, 53, 348—350

Фокусирующая система 40
 — — электромагнитная 40
 — — электростатическая 40
 Фонема 345, 373
 Фонетические правила 354—357
 Формантная частота 348
 Фотонаборная установка 81, 82

Цифро-аналоговый преобразователь
 (ЦАП) 41, 292—300
 Цифровой дифференциальный анали-
 затор (ЦДА) 28—35
 — для генерации окружностей 33—
 35
 — простой 30, 31
 — симметричный 28—30

Шина адресная 252
— данных 252
— информационная 55
— мультиплексируемая 290
— управляющая 322
Шрифтоноситель 77

Экстраполяция 365
— линейная 365—368

— с изменяемым опорным напряжением 367
Электронная пушка 38—40
— — записывающая 46
— — проявляющая 46
Электронно-лучевая трубка (ЭЛТ)
II, 38—47
— запоминающая 45—47
— с регенерацией изображения 54
— — теневой маской 43
— — сквозным пучком 42

**THE
McGRAW-HILL
COMPUTER
HANDBOOK**

**Editor in Chief
Harry Helms**

**Overview By
Adam Osborne**

**Foreword by
Thomas C. Bartee**

**McGraw-Hill Book Company
New York St. Louis San Francisco Auckland
Bogotá Hamburg Johannesburg London Madrid
Mexico Montreal New Delhi Panama Paris
São Paulo Singapore Sydney Tokyo Toronto
1983**

ББК 32.97
К 63
УДК 681.3

Перевод выполнен В. В. Пржиялковским, И. П. Пчелинцевым, М. В. Сергиевским, А. В. Чукашевым, А. В. Шалашовым, Т. А. Шаргиной, В. С. Штаркманом

Виатровски К., Гивоне Д., Коперда Ф., Мясковски С., Ньюмен У., Россер Р., Спрулл Р., Стоут Д., Хаус Ч., Хелмс Г., Хюэнштейн Л.

Компьютеры: Справочное руководство. В 3-х т. Т. 3. Пер. К63 с англ./Под ред. Г. Хелмса — М.: Мир, 1986. — 403 с., ил.

В третьем томе рассматриваются технические средства машинной графики, основные типы печатающих устройств и графопостроителей. Излагаются принципы построения микропроцессоров, методы программирования и создания операционных систем, а также вопросы сопряжения микроЭВМ. Значительное внимание уделено методам синтеза и распознавания речевых сигналов с помощью микропроцессорных систем.

Для инженеров, работающих с вычислительной техникой, аспирантов и студентов соответствующих специальностей вузов.

К 2405000000-086
041(01)-86 173-86, ч. 1

ББК 32.97
6Ф7

Редакция литературы по информатике и электронике

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Ваши замечания о содержании книги, ее оформлении, качестве перевода и другие просим присылать по адресу: 129820, Москва, И-110, ГСП, 1-й Рижский пер., 2, издательство «Мир».

Клод Внатровски, Дональд Гивоне, Фрэнк Коперда, С. Мясковски, У. Ньюмен,
Р. Россер, Р. Спрулл, Д. Стоут, Ч. Хаус, Г. Хелмс, Л. Хознстейн

**КОМПЬЮТЕРЫ.
СПРАВОЧНОЕ РУКОВОДСТВО.
ТОМ 3**

Старший научный редактор Серегина Н. В. Младший научный редактор Григоренко М. Ю.
Художник Бычков С. А. Художественный редактор Иванов Н. М.
Технический редактор Кренделева И. М. Корректор Денисова С. А.

ИБ № 5501

Сдано в набор 11.05.85. Подписано к печати 05.11.86.
Формат 60×90^{1/16}. Бумага ки.-журн. № 1. Печать высокая. Гарнитура латинская,
Объем 12,75 бум. л. Усл. печ. л. 25,50. Усл. кр.-отт. 25,50. Уч.-изд. л. 24,35.
Изд. № 41/4004. Тираж 50 000 экз. Зак. 665. Цена 1 р. 50 к.

ИЗДАТЕЛЬСТВО «МИР». 129820, ГСП, Москва, И-110, 1-й Рязский пер., 2.

Ленинградская типография № 2 головное предприятие ордена Трудового Красного Знамени Ленинградского объединения «Техническая книга» им. Евгении Соколовой Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 198052, г. Ленинград, Л-52, Измайловский проспект, 29.